KATHOLIEKE
UNIVERSITEIT
LEUVEN

DEPARTEMENT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN

RESEARCH REPORT 0225

PROJECT SCHEDULING UNDER UNCERTAINTY
SURVEY AND RESEARCH POTENTIALS

by
W. HERROELEN
R. LEUS

D/2002/2376/25

Project Scheduling Under Uncertainty Survey and Research Potentials[§]

Willy Herroelen and Roel Leus

May 2002

Operations Management Group
Department of Applied Economics
Katholieke Universiteit Leuven
Naamsestraat 69, B-3000 Leuven (Belgium)
Phones +32 16 32 69 70 and +32 16 32 69 67
Fax +32 16 32 67 32
e-mail: <first name>.<name>@econ.kuleuven.ac.be

[§] Invited paper to be published in the special issue of the *European Journal of Operational Research* that will contain selected papers presented at *PMS*2002, the *Eighth International Workshop on Project Management and Scheduling*, April 3-5, 2002, Valencia, Spain.

Project Scheduling Under Uncertainty – Survey and Research Potentials

Willy Herroelen and Roel Leus[‡]

ABSTRACT

The vast majority of the research efforts in project scheduling assume complete information about the scheduling problem to be solved and a static deterministic environment within which the pre-computed baseline schedule will be executed. However, in the real world, project activities are subject to considerable uncertainty, that is gradually resolved during project execution. In this survey we review the fundamental approaches for scheduling under uncertainty: reactive scheduling, stochastic project scheduling, stochastic GERT network scheduling, fuzzy project scheduling, robust (proactive) scheduling and sensitivity analysis. We discuss the potentials of these approaches for scheduling projects under uncertainty.

Keywords: project management and scheduling; scheduling under uncertainty; robustness; schedule stability

[‡] Research Assistant of the Fund for Scientific Research, Flanders (Belgium) (F.W.O.)

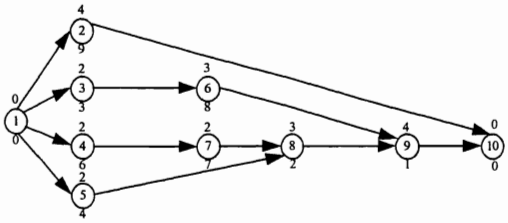
1. Introduction

The project scheduling literature largely concentrates on the generation of a precedence and resource feasible schedule that "optimises" the scheduling objective(s) (most often the project duration) and that should serve as a baseline schedule for executing the project. Such a *baseline schedule* (also called a *predictive schedule* or *pre-schedule*) serves very important functions (Mehta and Uzsoy, 1998). The first is to allocate resources to the different activities to optimise some measure of performance. The second, as also pointed out by Wu et al. (1993), is to serve as a basis for planning external activities such as material procurement, preventive maintenance and delivery of orders to external or internal customers. Baseline schedules serve as a basis for communication and coordination with external entities in the company's inbound and outbound supply chain. Based on the baseline schedule, commitments are made to subcontractors to deliver materials, support activities are planned (set-ups, supporting personnel), and due dates are set for the delivery of project results. Moreover, from the modelling viewpoint, many real-life scheduling problems such as course scheduling, sports time-tabling, railway and airline scheduling, can be modelled as variations of resource-constrained project scheduling problems. In these environments executing activities according to the pre-schedule is a must that is imposed by the customer: although "technically" possible, activities may not start prior to their scheduled starting time.

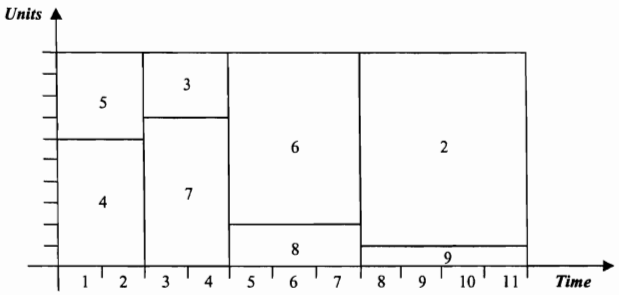
During project execution, however, project activities are subject to considerable uncertainty that may lead to numerous schedule disruptions. This uncertainty may stem from a number of possible sources: activities may take more or less time than originally estimated, resources may become unavailable, material may arrive behind schedule, ready times and due dates may have to be changed, new activities may have to be incorporated or activities may have to be dropped due to changes in the project scope, weather conditions may cause severe delays, etc. A disrupted schedule incurs higher costs due to missed due dates and deadlines, resource idleness, higher work-in-process inventory and increased system nervousness due to frequent rescheduling. As a result, the validity of static deterministic scheduling has been questioned and/or heavily criticised (Goldratt, 1997).

Uncertainty lies at the very heart of project management. A baseline schedule that is determined to be optimal with regard to some objective function prior to its execution may be very vulnerable to minor or serious disruption. As an illustration, consider the project shown in Figure 1(a) in activity-on-the-node format (Wiest and Levy, 1977). The project consists of eight real activities (activities 1 and 10 are dummies with zero duration). The duration of an activity is shown above the corresponding node. The number shown below a node is the constant per period requirement for a single renewable resource. The precedence relations are of the finish-start type with zero time-lag. Figure 1(b) shows a minimum duration baseline schedule (the project duration equals the critical path length) that yields a perfectly levelled resource profile with a constant per period resource requirement of 10 units. The schedule, however, is extremely vulnerable to uncertainty. The true optimality of the schedule can only be ascertained in conjunction with its execution in the real world. The slightest delay in the starting time of an activity, and/or the slightest increase in the duration of any activity, for

example, will lead to an immediate increase in the project makespan. The baseline schedule, determined to be optimal prior to its execution, clearly has insufficient built-in flexibility for dealing with unexpected events; in other words, is not "robust".



(a) Project network



(b) Resource profile

Figure 1. Project network and optimal baseline schedule (adapted from Wiest and Levy, 1977)

In general, we may distinguish between six approaches to dealing with uncertainty in a scheduling environment: reactive scheduling, stochastic scheduling, GERT network scheduling, scheduling under fuzziness, proactive (robust) scheduling, and sensitivity analysis. In this paper we will discuss these approaches mainly from a project scheduling viewpoint. In those situations where the approaches were clearly conceived in a machine scheduling context, our aim is to reveal their potentials for scheduling projects under uncertainty.

Reactive scheduling does not try to cope with uncertainty in creating the baseline schedule, but revises or re-optimises the baseline schedule during project execution when an unexpected event occurs.

Stochastic project scheduling does not create a baseline schedule but views the problem of scheduling projects under precedence and resource constraints as a multi-stage decision process which uses so-called scheduling policies (or scheduling strategies) that dynamically make scheduling decisions at stochastic decision points t, based on the observed past and the a priori knowledge about the activity processing time distributions.

The common objective considered in the literature is to create a policy that minimizes the expected project duration.

Stochastic project networks (GERT networks) have been introduced to deal with projects with stochastic evolution structure and feedback. A GERT network is an activity-on-the-arc network where the arcs (i,j) are assigned a weight vector (p_{ij}, F_{ij}). $p_{ij} > 0$ is the conditional execution probability of the corresponding activity (i,j) given that project event i has occurred. F_{ij} is the conditional distribution function of the non-negative duration $\boldsymbol{d}_{ij}$ of activity (i,j) given that (i,j) is carried out. GERT networks possess six different node types resulting from the combination of three possible entrance sides and two exit sides of a node. Whereas the temporal analysis of GERT networks has been studied quite extensively, GERT network scheduling, i.e. scheduling the activities in the presence of resource (machine) constraints in order to minimize an objective function in expectation (e.g. minimize the expected makespan) has only recently made its appearance in the literature.

The advocates of *fuzzy project scheduling* argue that probability distributions for the activity durations are unknown due to the lack of historical data. As activity durations are estimated by human experts, often under a non-repetitive (if not unique) setting, project management is often confronted with judgmental statements that are vague and imprecise. In those situations, which involve imprecision rather than uncertainty, the fuzzy set scheduling literature recommends the use of fuzzy numbers for modelling activity durations, rather than stochastic variables.

Proactive or *robust scheduling* aims at the construction of a baseline schedule that takes into account information about the uncertainty, for instance information about the variability in activity durations. In other words, a robust schedule is protected against uncertain events that occur during project execution. In addition to these approaches, recent efforts have emerged in the area of *sensitivity analysis* for polynomially solvable and intractable scheduling problems.

The paper is organised as follows. In the next section, we survey the research efforts in the field of reactive scheduling. In Section 3 we present a classification scheme for schedule construction techniques under uncertainty. Stochastic project scheduling is discussed in Section 4. Section 5 focuses on GERT network scheduling. Section 6 is devoted to fuzzy project scheduling. In Section 7 we characterize robust baseline schedules and review various robustness/stability measures as well as methods for generating robust and stable schedules that may have potential application for scheduling projects under uncertainty. Sensitivity analysis is discussed in Section 8. Overall conclusions and suggestions for further research conclude the paper.

2. Reactive scheduling

Reactive scheduling does not try to cope with uncertainty in creating the baseline schedule, but revises or re-optimises the baseline schedule when an unexpected event occurs. Basically most efforts concentrate on "repairing" the baseline schedule to take into account the unexpected events that have come up. For a review of the extensive literature in shop environments we refer to Sabuncuoglu and Bayiz (2000) and Selke and Kerr (1994).

The reactive scheduling action may be based on various underlying strategies. At one extreme, the reactive effort may rely on very simple techniques aimed at a quick schedule consistency restoration. We shall refer to these approaches as *schedule repair* actions. A typical example of such a simple control rule is the well-known *right shift rule* (Sadeh et al., 1993; Smith, 1994). This rule will move forward in time all the activities that are affected by the schedule breakdown because they were executing on the resource(s) causing the breakage or because of the precedence relations. It should be clear that this strategy may lead to poor results as it does not re-sequence activities.

At the other extreme, the reactive scheduling approach may involve a full scheduling pass of that part of the project that remains to be executed at the time the reaction is initiated. Such an approach will be referred to as *(full) rescheduling* and may use any deterministic performance measure, such as the new project makespan. In a sense, schedule repair is a heuristic rescheduling pass. If our objective would be to generate a new schedule that deviates from the original schedule as little as possible, we are in the particular rescheduling case where we want to induce *ex post stability* (the ex ante case will be discussed in the next section). Such a *minimum perturbation strategy* may rely on the use of exact and suboptimal algorithms using as objective function the minimization of the sum of the (weighted) absolute differences between the start time of each activity in the repaired schedule and the original start time of that activity (El Sakkout et al., 1998). A related approach is to *match-up* with the pre-schedule at a certain time in the future, whenever a deviation from the initial parameter values (mainly deviations from the activity duration projections) arises (Bean et al., 1991; Wu et al., 1993; Akturk and Gorgulu, 1999; Alagoz and Azizoglu, 2001).

Artigues and Roubellat (2000) study the case where, in a multi-project, multi-mode setting with ready times and due dates, it is desired to *insert a new unexpected activity into a given baseline schedule* such that the resulting impact on maximum lateness is minimized. The authors perform a clever rescheduling pass in which they restrict the solution to those schedules in which the resource allocation remains unchanged. Using a resource flow network representation (see section 7.1.3.2.2) they develop a stepwise procedure for generating a set of dominant insertion cuts for the network. From each dominant insertion cut, they then derive the best execution mode and valid insertion arc subset included in the dominant insertion cut. The authors have validated their polynomial insertion algorithm on the 110 Patterson test problems (Patterson, 1984) against complete rescheduling using the MINSLACK priority rule within a serial schedule generation scheme. In terms of computational burden the insertion method clearly outperforms the rescheduling method. The mean increase of the makespan of the schedule with the inserted activity above the makespan of the baseline schedule stays below the duration of the activity to be inserted. Moreover, the mean makespan increase is smaller for the insertion algorithm.

3. Generating a baseline schedule

The first column of Table 1 distinguishes between three basic approaches for the development of a baseline schedule. In the first approach, *no baseline schedule* is generated. Apart from its basic functions mentioned in the introduction, a baseline schedule will be useful when variability, for example in the activity durations, is not

purely stochastic but rather 'manageable' to a certain extent, such that management actions can be undertaken to 'stick' to the schedule. This will be more applicable when resources are human beings and/or when overtime options exist, and less when resources are mainly machines and no activity speed-up can be enforced. In the second scheme, a baseline schedule is developed using a deterministic scheduling method *without any anticipation of variability* in the input parameters that may occur during project execution. Single point estimates are used for parameters such as activity durations. The third approach is to develop a baseline schedule that *incorporates a degree of anticipation of variability* during project execution. This approach may use information about the particular variability characteristics (for example probability distributions for activity durations) and/or information about the reactive scheduling approach that will be adhered to during project execution (mostly very simple repair operations). This setting will be referred to as *proactive* or *robust* scheduling. The special case where the baseline objective is to minimize a function of the deviation between the baseline and the final schedule, focuses on *ex ante stability*. Often the term *quality robustness* is used when referring to the sensitivity of the schedule performance in terms of the objective value, while the term *stability* or *solution robustness* is used to refer to the insensitivity of the activity start times to changes in the input data. Robustness is closely related to *flexibility* (Sörensen, 2001). A schedule is called flexible if it can be easily repaired, i.e. changed into a new high quality schedule. The informal French association of researchers in scheduling GOThA (Groupe de recherche en Ordonnancement Théorique et Appliqué – http://www-poleia.lip6.fr/~sourd/gotha/) has established a "Flexibility working group" that regularly reflects on how to define, measure and use flexibility and also maintains a web page listing recent references (http://www.loria.fr/~aloulou/pages/biblio_gotha.html).

A second distinction can be made with regard to the way in which decisions will be taken during project execution on how to react to disruptions and when to start new activities. Three possibilities are listed in the second column of Table 1: (i) when a baseline exists, we reschedule, using any of the options that were discussed in the previous section; (ii) before the start of the project, a scheduling policy is chosen that will determine how to act during schedule execution (when a baseline is used, we will exclude the 'trivial' scheduling policy that corresponds with rescheduling (in any form) and assume that a number of essential decisions are made before the start of the project), and (iii) instead of using preset scheduling policies, project management makes decisions as the project develops. Any combination of the approaches in the first and second column of Table 1 may be used, except of course for rescheduling without a baseline.

Table 1. Different methods for schedule generation under uncertainty.

baseline schedule	*during project execution*
no baseline schedule	rescheduling
no anticipation of variability	scheduling policies
with anticipation of variability	management decisions

Apart from these methods for construction of the final schedule, algorithms have also been devised to provide the project manager with information about *allowable* deviations in project parameters, which will aid the manager in determining which parts

of the project require the most attention (the inherent assumption is that the sources of uncertainty are more or less manageable). Sensitivity analysis, to be discussed in Section 8, is a clear example of such an approach.

4. Stochastic project scheduling

4.1 *Stochastic resource-constrained project scheduling*

The literature on stochastic project scheduling is rather sparse (for a detailed discussion, see Chapter 9 in Demeulemeester and Herroelen (2002)). Most efforts concentrate on the so-called *stochastic resource-constrained project scheduling problem*, i.e., the problem of scheduling project activities with uncertain durations in order to minimize the expected project duration subject to zero-lag finish-start precedence constraints and renewable resource constraints. The project is represented by an activity-on-the-node network $G=(V,E)$, where the set $V=\{1,2,...,n\}$ denotes the set of activities. Activity 1 and n are dummy activities, representing the start and end of the project. The durations of the other activities are given by a random vector $\boldsymbol{d}=(\boldsymbol{d}_2,\boldsymbol{d}_3,...,\boldsymbol{d}_{n-1})$, where $\boldsymbol{d}_i$ denotes the random duration of activity i. We denote a particular *realization* or *sample* of $\boldsymbol{d}$ as $d=(d_2,d_3,...,d_{n-1})\in\Re_+^n$. The arcs of set E define the zero-lag finish-start precedence relations among the activities. The renewable resources $(k=1,2,...,K^\rho)$ are available in constant integer amounts a_k^ρ. The non-dummy activities require an amount of $r_{ik}^\rho \le a_k^\rho$ units of renewable resource type k. Given the presence of both resource constraints and random activity durations, schedules are generated through the application of so-called *scheduling policies* or *scheduling strategies*, and no baseline schedule is used.

According to the definitions given in Igelmund and Radermacher (1983ab) and Möhring et al. (1984,1985), a *scheduling policy* Π makes decisions at the *decision points* $t=0$ (the *start of the project*) and the *completion times of activities*. A decision at time t is to start at time t a precedence and resource feasible set of activities, $S(t)$, exploiting only information that has become available up to time t. As soon as the activities have been finished, the activity durations are known yielding a realization d of activity durations. The application of policy Π leads to the creation of a schedule $\Pi(d)=(s_1,s_2,...,s_n)$ of activity starting times and a resulting schedule makespan $C_{\max}(\Pi(d))$. The common objective considered in the literature is to create a policy that minimizes the expected project duration $E(C_{\max}(\Pi(\boldsymbol{d})))$ over a class of policies. Fernandez (1995), Fernandez et al. (1996) and Pet-Edwards et al. (1998) show how to write the corresponding optimisation problem in its general form as a multi-stage stochastic programming problem.

A complete characterization of all policies and corresponding subclasses can be found in Möhring et al. (1984, 1985). A well-known class of scheduling policies is the class of *priority policies* which order all activities according to a *priority list* and, at every decision point t, start as many activities as possible in the order dictated by the list. The deterministic equivalent of such a policy is often denoted as *list scheduling*. List scheduling policies share a number of drawbacks. First of all, priority policies cannot

guarantee an optimal schedule. Moreover, they share the disadvantage that the so-called Graham anomalies may occur. These anomalies have been described by Graham (1966) in the context of parallel machine scheduling. For example, decreasing activity durations may lead to a makespan increase, adding additional capacity (machines) may result in an increase in the schedule makespan and removing precedence constraints may also lead to a makespan increase. A large amount of research has been devoted to the development of early start policies and pre-selective policies, which do not suffer from the undesirable Graham anomalies.

Radermacher (1985) describes *early start* (*ES*) *policies* using the concept of minimal forbidden sets. *Minimal forbidden sets* are inclusion minimal sets of pairwise not precedence related activities that cannot be scheduled simultaneously because they share limited resources. 'Inclusion minimal' means that each proper subset of a forbidden set may be executed simultaneously without violating any resource constraints. The number of forbidden sets may grow exponentially in the number of activities. A policy Π is an *ES*-policy if for each minimal forbidden set F there exists a pair $(i,j), i,j \in F, i \neq j$, such that for each sample d of activity durations, j cannot be started before i has finished. *ES*-policies can easily be implemented by adding the pairs (i,j) to the original set of precedence relations and computing the earliest activity start times as $s_1 = 0$ (starting dummy) and $s_j = \max_{(i,j)\in E}(s_i + d_i), j \in V \setminus \{1\}$.

Igelmund and Radermacher (1983ab) introduced *pre-selective* (*PRS*) *policies*. A policy Π is pre-selective if for each minimal forbidden set F there exists an activity $j \in F$ (the *preselected* or *waiting activity*), such that for each sample d of activity durations, j is not started before some activity $i \in F \setminus \{j\}$ has finished. A *selection* is a sequence of waiting activities for all minimal forbidden sets. Möhring and Stork (2000) have introduced a very useful representation of pre-selective policies using so-called *waiting conditions*. Waiting conditions can be modelled as AND/OR precedence constraints (Gillies and Liu (1995), Möhring et al. (2000)). A waiting condition is given by a pair $(X,j), X \subset V, j \in V \setminus X$, where activity j cannot be started before at least one activity $i \in X$ has finished. Each restriction imposed by a minimal forbidden set F and its preselected activity j can be represented by the waiting condition $(F \setminus \{j\}, j)$. Obviously, each given precedence constraint $(i,j) \in E$ can be represented by the waiting condition $(\{i\}, j)$. A set W of waiting conditions induces a digraph D which contains a node for each activity and for each waiting condition (X,j), directed arcs (i,w) are included for each $i \in X$, with w the node representing (X,j), as well as an extra arc (w,j) (Stork 2000).

We alert the reader to the fact that pre-selective policies do have severe computational limitations. First of all, the number of forbidden sets may be exponential in the number of activities. Since a pre-selective policy is defined by a preselected activity for each forbidden set, exponential storage space is needed for storing the selection defining a policy. This handicap becomes a serious problem when pre-selective policies have to be enumerated within branch-and-bound algorithms. Second, many pre-selective policies are dominated by others, while there is no good algorithm to test dominance among pre-selective policies. These drawbacks inspired Möhring and Stork (2000) to define linear pre-selective policies. *Linear pre-selective policies* (*LIN*) are a subclass of the class of pre-selective policies. The authors define a selection by a priority ordering L of the activities (respecting the original precedence constraints) in such a way

that the preselected waiting activity of the minimal forbidden set F is the activity with the smallest priority, i.e., the last activity in the list L.

A policy is *job based* (*JBP*) if it is linear pre-selective (according to some ordering L of the activities) and if $s_i \leq s_j$ for each sample d and for $i \prec_L j$. For a given sample d, the earliest activity start times can be computed by starting each activity in the order imposed by L as early as possible but not earlier than the start time of some previously started activity. Clearly, the job-based policies use an "activity based" point of view and not a "resource based" view. As a result, job-based policies do not require the use of the forbidden sets. This is a very efficiency gaining characteristic since activity based policies can easily be applied to very large projects for which the number of forbidden sets may be exorbitant.

4.1.1 Branch-and-bound

Stork (2000) has implemented branch-and-bound algorithms to compute optimal *ES*-, *PRS*-, *LIN*- and *JBP*-policies, using two branching schemes, lower bound calculation and various dominance rules. He validates the algorithms on the well-known 480 test instances generated using the problem generator *ProGen* (Kolisch and Sprecher, 1996). Each instance consists of 20 activities which require at most 4 different renewable resources. The average number of forbidden sets is roughly 70 (maximum 774). He takes the given deterministic activity durations for each activity as expected value and constructs uniform and triangle, as well as approximately normal, Gamma and exponential distributions. The 200 samples d from $\boldsymbol{d}$ are then generated using standard simulation techniques assuming independent activity durations.

The algorithms are coded in C++ and are run on a 143 MHz Sun Ultra 1 machine. 107 instances could be solved by all algorithms within 1000 seconds of CPU time and 50 MB of main memory. The best algorithm for *PRS*-policies solves 390 out of the 480 instances in an average CPU time of roughly 70 seconds for all considered types of distributions. The enumeration of *JBP*-policies is extremely time intensive. Only 161 out of 480 instances were solved to optimality. Computer memory caused problems to the *ES*-policies (87 instances exceeded the limit of 50 MB).

Pre-selective policies yielded the smallest expected makespan among all considered classes of policies, which is logical because the set of *PRS*-policies embraces all *LIN*- and *JBP*-policies, and clearly dominates the *ES*-policies. The other policy classes yield values that are at most 0.5 worse on average (maximally 2.1%). On average, the expected makespan was more than 4% larger than the deterministic makespan, with the maximal percentage deviation occasionally being greater than 10%.

Restricting the running time to 100 seconds, Stork (2000) also reports on results obtained on 480 *ProGen* instances with 30 and 60 activities, respectively. For *LIN*-policies, 179 out of the 480 30-activity instances can be solved to optimality. Moreover, on average over all 480 instances, the obtained *LIN*-policies produced the best feasible solutions, although the optimal value for *PRS* will never be worse than *LIN*. Almost none of the 60-activity instances were solved to optimality. If a large number of forbidden sets makes the use of linear selective policies impossible, Stork (2000) recommends the use of job-based policies to generate feasible solutions of acceptable quality.

4.1.2 Heuristic procedures

Research on heuristic procedures for solving the stochastic RCPSP is just emerging (Pet-Edwards (1996), Golenko-Ginsburg and Gonik (1997), Tsai and Gemmill (1996, 1998)). As an illustration, we briefly discuss the procedures of Golenko-Ginsburg and Gonik (1997) and the tabu search procedure of Tsai and Gemmill (1998).

Golenko-Ginsburg and Gonik (1997) consider PERT type activity-on-the-arc networks where the duration of an activity is a random variable with given density function (beta, uniform and normal distributions are used) and where a pre-given lower and upper bound on the activity duration is available. The activities require a constant amount of renewable resources during their execution. The renewable resources are available in constant amounts throughout time. The objective is to minimize the expected project duration.

The basic logic of the algorithm is as follows. At each activity completion time t, the algorithm computes for each unscheduled activity the probability p_j that activity j is on the critical path when all resource conflicts occurring after time t are neglected. The conditional probabilities p_j are approximated using simulation. At each decision point t, all the activity durations for the unscheduled activities are simulated using one of the alternative density functions (normal, uniform, beta). Then the critical path of the remaining network with simulated activity durations is determined. By repeating this procedure many times, frequencies are obtained for each activity to be on the critical path. These frequencies are taken as the p_j. After obtaining the probabilities for all the competing precedence-feasible eligible activities E at time t, a subset $E' \subseteq E$ of activities is started at t with the property that $\sum_{j \in E'} r_{jk}^{\rho} \leq a_k^{\rho}$ for all renewable resources k and the sum of the activity contributions of the activities $j \in E'$ is maximized. For each activity j, its contribution is the product of its probability p_j of lying on the critical path and its average duration. The authors suggest to solve this multi-dimensional knapsack problem using 0-1 programming. In addition they provide a heuristic procedure that schedules the competing activities in descending order of their contribution (if the conditional probabilities are zero, in descending order of the mean duration).

The authors provide computational results for an example project with 36 activities and requirements for a single renewable resource. The instance has 3730 minimal forbidden sets. They compute a feasible solution with an expected makespan of 448.85, 448.49 and 433.88 for, respectively, the normal, uniform and beta distribution case using the 0-1 programming approach. The heuristic finds an expected makespan of 461.58, 461.35 and 447.98. The authors do not report on running times for the procedures.

Stork (2000) reports computational results on the same instance using the uniform distribution case. The initial activity-based priority policy yielded already an initial upper bound of (rounded) 445 within negligible computation time. The activity-based priority policy in combination with the precedence tree branching scheme found a solution with an expected project duration of (rounded) 434 in less than 40 seconds. The other algorithms were not able to improve their initial upper bound solution within a time limit of 100 seconds.

Tsai and Gemmil (1998) report computational results for the well-known 110 Patterson test problems using a tabu search algorithm. They assume a beta distribution to model activity durations and use an optimistic, most likely and pessimistic time estimate to calculate the parameters of the beta distribution. Solutions again correspond with activity priority lists.

Using the expected activity durations, they compute an initial feasible solution using the minimum slack rule. The expected project duration of a feasible solution is computed as follows: (a) a duration for each activity is drawn from the beta distribution with the parameters calculated using the three time estimates, (b) given the feasible sequence and the randomly generated activity durations, the project duration is computed, (c) the calculation of the project duration is repeated 100 times and then the average project duration for the particular feasible sequence is reported as the expected project duration. It should be noted that this approach to estimate the expected project duration violates the so-called *non-anticipativity constraint* (Fernandez et al., 1996). The approach implicitly assumes that all uncertainty with regard to activity durations is resolved before the start of project execution ('anticipative'), which will only rarely be the case. Rather, information will normally become available only gradually as time progresses, making the use of scheduling policies, as described above, more appropriate.

The structure of the tabu search algorithm developed by the authors is classical and rather straightforward. The procedure computes a list of candidate moves by randomly selecting two activities to switch positions. If the generated sequence is not feasible, two new activities are selected. The procedure is repeated until a pre-specified number of moves are found. Each candidate move is evaluated by computing its expected project duration. If the move yields a better expected project duration than all other moves found admissible so far in the candidate list, the tabu status of the move is checked. Two tabu lists are used: one containing critical activities, the other containing non-critical activities. A move is labeled tabu restricted if the activity moved back in time is a critical activity in the critical tabu list or the activity moved forward in time is a non-critical activity in the non-critical tabu list. If the move is not tabu restricted it is accepted, otherwise it is the subject of a simple aspiration test. The aspiration test used by the authors is also straightforward: if the project duration of the new sequence is shorter than the best project duration found thus far, the tabu status is overruled.

For the randomized Patterson test instances, the authors use the known deterministic optimal makespan times 1.05 as an approximate lower bound (the expected activity durations are 1.05 times the deterministic durations). Almost one-half of the instances have their duration decreased over 10%, and some of the projects have their duration decreased by over 20% from the initial solutions obtained using the minimum slack heuristic. The average increase above the approximate lower bound was around 3% with an average execution time of slightly more than 11 seconds.

4.2 *Stochastic activity interruptions*

Valls et al. (1999) have studied the problem of scheduling resource-constrained project activities that are either deterministic (i.e. have a known duration and cannot be interrupted) or stochastic (i.e. may be interrupted for an uncertain amount of time and resumed later). The initial processing time d_{i1} of an activity i that may be interrupted is

assumed to be known with certainty, however, the length of the interruption w_i and the remaining processing time after the interruption d_{i2} are uncertain. An example of such a situation may be a project in which some activities are submitted to an approval process before they can be completed. The time to review and approve the work performed during the initial processing determines the length of the interruption, while the outcome of the approval process may determine the length of the final processing. Each activity has a due date δ_i and a tardiness penalty c_i. Each activity requires a constant per period amount of a renewable resource during its execution. The renewable resource types are available in a constant per-period amount. The two parts of an interrupted activity require the same number of units from each resource. The processing time of the second part d_{i2} of an interrupted activity i is independent of the length of the interruption w_i. The objective is to schedule the activities subject to the zero-lag finish-start precedence constraints and the resource constraints in order to minimize the expected total weighted tardiness.

The authors have developed a scenario-based approach. The scenarios are generated by specifying three time estimates both for the interruption and the second part of each stochastic activity. The solution algorithm is a hybrid algorithm based on the scatter search methodology. In step 1, heuristic priority rules are used in combination with a parallel scheduling scheme to generate a starting set of solutions (so-called trial points). A solution is obtained using a two-stage decision problem: in the first stage a priority is assigned to each activity and in the second stage, once the uncertainty about the interruptions is resolved, a schedule is constructed using these priorities. A subset of the best solutions is selected to serve as reference points. In step 2, clustering strategies that allow for diversification and intensification (Glover and Laguna, 1997) are used to form structured combinations of subsets of the current reference points to create new points. In step 3, a collection of the best points generated in step 2 are extracted to be used as starting points for a new application of the heuristics in step 1. The steps are repeated until a specified iteration limit is reached.

The authors report on computational results obtained on a set of randomly generated test problems that demonstrate that the scatter search procedure is robust (good solutions can be obtained using a relatively small set of scenarios) and effective (a significant reduction is obtained in the average objective function value obtained by the heuristics used to generate the initial population). The authors have extended the approach to the problem of minimizing the weighted tardiness of jobs with stochastic interruptions in a parallel machine environment (Laguna et al., 2000).

4.3 *The stochastic discrete time/cost trade-off problem*

The literature on the stochastic version of the discrete time/cost trade-off problem is virtually void. Wollmer (1985) discusses a stochastic version of the deterministic linear time/cost trade-off problem for activity-on-the-arc networks in which the duration of an activity can be described as $y_{ij} + \xi_{ij}$, where the decision variable y_{ij} is bounded from below by the activity crash duration l_{ij} and is bounded from above by the normal duration of the activity u_{ij}. ξ_{ij} is a bounded discrete random variable, independent

of y_{ij} with an expected value of 0. Each activity (except dummies of course) has an associated non-negative cost c_{ij}, which is the cost per unit decrease in y_{ij} within the range of l_{ij} and u_{ij}. The objective then is to determine activity durations y_{ij} and event realization times which minimize the expected project completion time subject to a budget constraint, or achieve a feasible fixed expected project completion time at minimum cost.

Gutjahr et al. (2000) describe a stochastic branch-and-bound procedure for solving a specific version of the stochastic discrete time/cost trade-off problem where so-called *measures* (like the use of manpower, the assignment of highly-skilled labour or the substitution of equipment) may be used to increase the probability of meeting the project due date and thus avoiding penalty costs. The authors assume that the duration of an activity (i,j) in an activity-on-the-arc network is modelled by a beta distributed random variable $\boldsymbol{d}_{ij}$. The distribution of each $\boldsymbol{d}_{ij}$ can be measured and the random variables $\boldsymbol{d}_{ij}$ are assumed to be independent. It is assumed that the distributions of the random variables $\boldsymbol{d}_{ij}$ might be changed by certain *crashing measures* $m = 1,...,M$. Typically, measure m reduces the expected time required for one or several activities by a certain amount. As such, the duration of activity (i,j) becomes dependent on the vector $x = (x_1, x_2, ..., x_M)$, where $x_m = 1$ if measure m is chosen and $x_m = 0$ otherwise. $\boldsymbol{d}_{ij}(x)$ will denote the duration of activity (i,j) on the condition that a measure combination described by the vector x has been chosen (in their experiments, the authors assign each measure randomly to an activity). Each measure m incurs an additional cost of c_m currency units. For each x, the project duration $C_{\max}(\boldsymbol{d}(x))$ can be computed on the basis of the values of $\boldsymbol{d}_{ij}(x)$ using standard critical path calculations. Since $C_{\max}(\boldsymbol{d}(x))$ depends on the stochastic durations $\boldsymbol{d}(x)$, it is also a random variable.

It is assumed that *penalty costs* occur if the project is completed after its pre-specified due date. These costs are described by a loss function Λ, where $\Lambda(t)$ is the loss occurring if the project finishes at time t. The authors assume that Λ is a step function that implies that no penalty occurs if the project is completed on time. The loss $\Lambda(C_{\max}(\boldsymbol{d}(x)))$ is also a random variable. The objective is to minimize the *expected* overall loss, which is equal to the crashing costs and the expected penalty costs. The authors report on computational results obtained on 33 random problem instances with 25, 50 and 100 nodes, beta distributed activity durations and 10, 15 or 20 crashing measures. CPU time limits are set to 2, 10 and 60 minutes, respectively, on a 133 MHz personal computer. The use of stochastic branch-and-bound in combination with deterministic branch-and-bound for solving the sub-problems could solve all the instances within the given runtime limit (the authors do not mention which deterministic branch-and-bound procedure is used). The best results were obtained using the heuristic procedure for solving the deterministic sub-problems and replacing the straightforward sampling with an importance sampling procedure.

4.4 *Multi-mode trade-off problems in stochastic networks*

At the time of writing, the literature on the stochastic multi-mode problem was virtually void. Jørgenson (1999) and Elmaghraby (2000) focus on a dynamic stochastic

resource allocation problem in activity-on-the-arc networks where an activity a requires total work content $W_a(k)$, a random variable, of resource $k = 1,...,K$ specified as renewable or nonrenewable over the entire planning horizon. An allocation of $x_a(k,t)$ units of resource k to activity a at time t costs $c_k(x_a(k,t), W_a(x_a))$ per unit of time, also a random variable. The resulting activity duration is denoted by the random variable $y_k(x_a) = g_k(W_a(x_a))$. The total activity cost is then the random variable $C_k(x_a) = c_k(x_a(k,t), W_a(x_a)).g_k(W_a(x_a))$. The project is assumed to have a fixed due date δ_n and a penalty function $p(t_n - \delta_n)$, where t_n is the random variable denoting the time of realization of node n. The penalty function is assumed to be linear with proportionality constant p_L; i.e. $p(t_n - \delta_n) = p_L.\max\{0, t_n - \delta_n\}$. The objective then is to determine the resource allocation vector X_a to all the project activities such that the total expected cost is minimized. In the case of nonrenewable resources, the objective is taken to be the minimization of the project duration.

Elmaghraby (2000) describes two dynamic programming models for solving the problem and illustrates them on a problem example. A new state space is introduced based on the concept of uniformly directed cutsets. For details, we refer the reader to Elmaghraby (2000). At the time of writing, computational results were not yet available.

Jørgenson (1999) and Elmaghraby (2000) demonstrate that the dynamic resource allocation approach is superior to static optimization that assumes certainty equivalents given by expected values. Deterministic static time/cost trade-off models underestimate the total expected project costs and neglect the value of flexibility. Updating the plans as new information becomes available by adjusting the amount of resources to be allocated may well lead to superior results. Computational experience in this area would be more than welcome.

5. GERT network scheduling

While the stochastic project scheduling approaches discussed in the previous section still assume that the evolution structure of the project network is specified in advance – each activity is carried out exactly once during a single project execution and it is not possible to return to previously performed activities – *stochastic project networks* (GERT networks) deal with projects with stochastic evolution structure. A GERT network is an activity-on-the-arc network with exactly one source node (the beginning event of the project occurring at time zero) and one sink node (the terminal project event). The network may contain cycles, allowing for the multiple execution of activities during the execution of the project. Each arc (i,j) is assigned a weight vector (p_{ij}, F_{ij}). $p_{ij} > 0$ is the conditional execution probability of the corresponding activity (i,j) given that project event i has occurred. F_{ij} is the conditional distribution function of the non-negative duration d_{ij} of activity (i,j) given that (i,j) is carried out. p_{ij} and F_{ij} are assumed to be independent of the number of times that project event i has occurred or activity (i,j) has been executed before, respectively (Neumann, 1984, 1990, 1999; Neumann and Steinhardt, 1979).

GERT networks possess six different node types resulting from the combination of three possible entrance sides and two exit sides of a node. A node has an AND

entrance if during project execution the node is activated (the corresponding project event occurs) when all incoming activities have been finished for the first time. A node has an inclusive-or (IOR) entrance if it is activated when one of the incoming activities has been finished for the first time. A node has an exclusive-or (EOR) entrance if it is activated every time when an incoming activity has been finished. The exit of a node is deterministic if all outgoing activities are carried out when the node has been activated. If exactly one outgoing activity is executed when the node has been activated, the node has a stochastic exit.

The treatment of GERT networks relies on several assumptions, the most important being that the durations of the different activities and of different executions of one and the same activity are independent. The temporal analysis of GERT networks has been studied quite extensively in the seventies and eighties (Neumann, 1984, 1990; Neumann and Steinhardt, 1979). These analytic methods, however, require a great computational effort. Therefore, simulation has been widely used for the evaluation of GERT networks (Pritsker, 1977, 1986; Pritsker and Sigal, 1983; Whitehouse, 1973). GERT *network scheduling*, the scheduling of GERT network activities in the presence of resource (machine) constraints in order to minimize project objective functions in expectation (such as minimising the expected makespan), has only received recent attention. A state-of-the-art survey of GERT network scheduling can be found in Neumann (1999). Forced by the state of the art, the author considers the resources to be machines and reviews methods for approximately solving single machine, parallel machine, job shop and flow shop problems with GERT network precedence constraints. The literature on resource-constrained project scheduling with GERT networks, however, is virtually void. The heavy computational burden of analytic treatment prohibits practical application and forces one to rely on simulation as the vehicle of analysis.

6. Fuzzy project scheduling

The advocates of the fuzzy activity duration approach argue that probability distributions for the activity durations are unknown due to the lack of historical data. As activity durations have to be estimated by human experts, often in a non-repetitive or even unique setting, project management is often confronted with judgmental statements that are vague and imprecise, for example: 'The duration of an activity is clearly more than 2 days and less than 5 days; about 3 days is usual'. In those situations, which involve imprecision rather than uncertainty, the fuzzy set scheduling literature recommends the use of fuzzy numbers for modelling activity durations, rather than stochastic variables. Instead of probability distributions, these quantities make use of *membership functions*, based on *possibility theory*.

A *fuzzy set* is a function that measures the degree of membership to a set. Set A in a base set X can be described by a *membership function* $\mu_A : X \rightarrow \{0,1\}$ with $\mu_A(x)=1$ if $x \in A$ and $\mu_A(x)=0$ if $x \notin A$. If it is uncertain whether or not element x belongs to set A, the above model can be extended such that the membership function maps into the interval [0,1]. A high value of this membership function implies a high possibility, while a low value implies a poor possibility. This leads to the definition of a *fuzzy set* $\tilde{A}$ in X as a set of ordered pairs $\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) | x \in X\}$, where $\mu_{\tilde{A}}(x)$, $0 \leq \mu_{\tilde{A}}(x) \leq 1$, is called the

membership function or *grade of membership* of x in $\tilde{A}$. In the classical case where $\mu_{\tilde{A}}(x) = 0$ or 1, $\tilde{A}$ is said to be a *crisp* set.

A *fuzzy number* $\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) | x \in X\}$, where $\mu_{\tilde{A}}$ is the membership function of $\tilde{A}$, is a special kind of a fuzzy set defined as a convex fuzzy subset of real line $\Re$, or $\forall a, b \in \Re, \forall c \in [a,b], \mu_{\tilde{A}}(c) \geq \min(\mu_{\tilde{A}}(a), \mu_{\tilde{A}}(b))$. It is also required that $\exists a \in \Re : \mu_{\tilde{A}}(a) = 1$. The advocates of fuzzy scheduling admit that the precise form of a fuzzy number is difficult to describe by an expert (Hapke et al., 1999). A practical way of getting suitable membership functions of fuzzy data has been proposed by Rommelfanger (1990). He recommends that the expert express his/her optimistic and pessimistic information about parameter uncertainty on some prominent membership levels by specifying intervals on $\Re$: the smallest interval $[\underline{m}, \overline{m}]$ for which $\mu(x) = 1$, meaning that x certainly belongs to the set of possible values; a larger interval $[\underline{m}^{\lambda}, \overline{m}^{\lambda}]$, containing $[\underline{m}, \overline{m}]$, for which it holds that values x have a good chance $\geq \lambda$ of belonging to the set of possible values; and a third interval $[\underline{m}^{\varepsilon}, \overline{m}^{\varepsilon}]$, containing the second, for which all values x have $\mu(x) < \varepsilon$. Values x with $\mu(x) < \varepsilon$ have a very small chance of belonging to the set of possible values; i.e. the expert is willing to neglect the corresponding values of x. Using a six-point representation, a fuzzy number $\tilde{M}$ is then represented by the list of symbols $\tilde{M} = (\underline{m}^{\varepsilon}, \underline{m}^{\lambda}, \underline{m}, \overline{m}, \overline{m}^{\lambda}, \overline{m}^{\varepsilon})$ as shown in Figure 2.

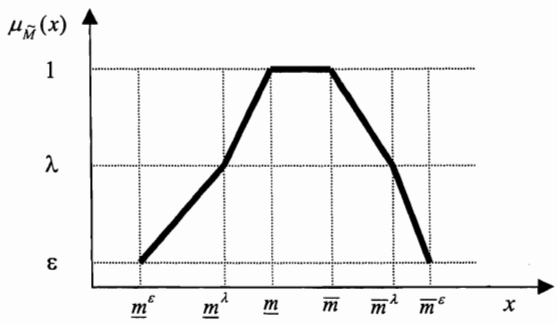


Figure 2. Fuzzy number $\tilde{M}$ in six-point representation (Hapke et al., 1999)

The output of a fuzzy scheduling pass will normally be a *fuzzy schedule*, which indicates fuzzy starting and ending times for the activities. Such fuzzy time instances may be interpreted as start or completion to a certain extent only. Lootsma (1997) gives the example of the last milestone of a construction project: during the opening of the new building, one will often notice loose ends behind curtains or under the carpets. The day after the opening ceremony, work is still in progress despite the official delivery, and the concept of 'completion' can be perceived to involve a degree of vagueness. In our view, a fuzzy schedule can also be viewed as a decision support tool rather than as a mere description of gradual completion and start of activities. As can be conceived from, amongst others, Dorn et al. (1995), a fuzzy schedule assists in the explicit representation of certain degrees of freedom in the predictive schedule, to represent the discretion

management has to start certain jobs a little earlier or later when duly propagating certain hard and soft constraints that may be imposed. In this sense, a fuzzy schedule comprises multiple crisp schedules.

The recent volume edited by Slowinski and Hapke (2000) gathers important recent work in fuzzy scheduling. At the time of writing, the literature on fuzzy resource-constrained project scheduling was in its burn-in phase (Hapke et al. (1994, 1999), Hapke and Slowinski (1996, 2000), Özdamar and Alanya (2000), Wang (1999).

The study of a fuzzy model of resource-constrained project scheduling has been initiated in Hapke et al. (1994) and Hapke and Slowinski (1996). They have extended the priority rule based serial and parallel scheduling schemes to deal with fuzzy parameters.

Hapke and Slowinski (2000) discuss the application of simulated annealing for solving the multi-objective fuzzy resource-constrained project scheduling problem. The procedure is an adaptation of the Pareto simulated annealing procedure developed by Czyzak and Jaskiewicz (1996) for solving crisp multi-objective combinatorial problems. The procedure has been incorporated in an integrated software package (Hapke et al., 2000). For details we refer to Hapke and Slowinski (2000).

Özdamar and Alanya (2000) study software development projects and offer a nonlinear mixed-binary mathematical problem formulation and accompanying solution heuristics. Their model incorporates uncertainties related to activity durations and network topology. The first type of uncertainty is attributed to error-prone coding that might result in elongated activity durations caused by validation and debugging sessions. Also, in practice, macro activities are often defined in order to simplify the planning of the project. Due to such a type of aggregation, it is more difficult to be precise on the duration of such a macro activity. The authors mention that a fuzzy logic approach to activity modelling will allow the project manager to be provided with a range of scenarios rather than a single one in the pre-planning phase. As a second motivation for using fuzzy numbers rather than stochastic variables, the authors note that we are dealing with subjective evaluations of human behaviour-related quantities. The authors use a six-point membership function, which allows for easy computation of the sum of the activity durations on a given path. The result of the maximum operator required in calculating early start times is approximated (overestimated) (Fortemps, 1997). Activities may be performed in one of different modes. Each mode m for activity i has a corresponding fuzzy duration $\tilde{d}_{im}$. Each different $\tilde{d}_{im}$ value has a corresponding six-point membership function μ_{im}. The authors use a first set of constraints to ensure that each activity is completed once in exactly one mode and assume that the members of the project team are modelled by renewable resources. In addition, there is an in-company consultant who is modelled as a continuous renewable resource who assigns part of his capacity to particular activities as the need arises. The uncertainty related to the network topology is due to common database design issues or program modules shared among parallel activities in the project network. This uncertainty is modelled by start-start precedence constraints with fuzzy time-lag. The objective function is to minimize the project duration. Özdamar and Alanya (2000) illustrate the use of four priority based heuristics: the standard minimum slack rule, the latest finish time rule, the maximum number of immediate successor rule and a minimum risk rule on a case study.

Wang (1999) has developed a fuzzy set approach to schedule product development projects having imprecise temporal information. The project has a fuzzy

ready time and fuzzy deadline and the activities are assumed to have a fuzzy duration, all described by a trapezoidal fuzzy number. The objective is to determine a start time for each activity such that the fuzzy ready time, deadline, precedence and resource constraints are satisfied. As the constraints are fuzzy, they are flexible in the sense that their satisfaction depending on the choice of start times are degrees within the range [0,1]. In addition, it may not be possible to find an acceptable schedule that 'partially' satisfies all the constraints. The author has developed a new method based on possibility theory to determine the satisfaction degrees for the constraints. A beam search procedure, based on the generation of groups of activities the delay of which resolves the resource conflicts (i.e. delaying alternatives (Demeulemeester and Herroelen, 2002), that selects only the most promising nodes at each level of the search tree (the so-called beam-width) for further expansion is developed to produce a set of fuzzy start times for each activity. Then, the crisp start time of each activity is determined based on possibility theory, to maximize the satisfaction degrees of all fuzzy constraints. This again illustrates our statement given above that a fuzzy schedule contains multiple crisp schedules, the choice between which is at the discretion of management. The author has validated the procedure against a fuzzy version of the A* algorithm (Bell and Park, 1990) on 30-, 60- and 90-activity problems taken from the PSLIB test library (Kolisch and Sprecher, 1996).

7. Proactive (robust) project scheduling

Numerous techniques for proactive (robust) scheduling have recently been published. The majority of publications are in the machine scheduling literature (Davenport and Beck, 2002). Various definitions for schedule *robustness* have appeared in the (machine) scheduling literature. Davenport et al. (2001) describe a robust schedule as a "schedule that is able to absorb some level of unexpected events without rescheduling". Jensen (2001) defines a robust schedule as a "quality schedule expected to still be acceptable if something unforeseen happens". Leon et al. (1994) define a robust schedule as "an *a priori* off-line schedule which maintains high performance in the presence of disruptions". More specifically their definition pertains to those job shop schedules for which a right-shift control policy is used that, on the occurrence of a disruption, maintains the scheduling sequence while delaying the unfinished jobs as much as necessary to accommodate the disruption. Le Pape (1991) defines a robust schedule as a "schedule with the ability to satisfy performance requirements predictably in an uncertain environment" and as a "schedule where the violation of the assumptions on which it is built are of no or little consequence". Daniels and Kouvelis (1995) and Kouvelis et al. (2000) view robust scheduling as "the determination of a schedule whose performance (compared to the associated optimal schedule) is relatively insensitive to the potential realizations of job processing times".

7.1 *Redundancy-based techniques*

A number of proactive techniques have been presented in the machine and project scheduling literature that aim at inserting some form of redundancy (extra time

and/or extra resources) in the schedule to absorb the disturbances caused by unexpected events during schedule execution.

7.1.1 Fault tolerant scheduling

Fault tolerance is a common practice in real-time pre-emptive single machine scheduling environments. Fault tolerance can be achieved through resource redundancy (multiple identical sets of resources kept in standby (Ghosh, 1996)) or time redundancy (scheduling of back-up tasks which simply reserve time for re-execution in the event of a fault (Ghosh et al., 1995)). Pure resource redundancy is rather unrealistic in a project environment: doubling the various resources would be cost prohibitive. Time redundancy may be relevant, but a (multi-) project environment is far off from the pre-emptive polynomially solvable single machine settings studied in a real-time environment.

7.1.2 Temporal protection

Temporal protection (Gao, 1995) extends the duration of activities based on the uncertainty statistics of the resources that are used for their execution. Resources that have a non-zero probability of breakdown are called *breakable resources*. The durations of activities requiring breakable resources are extended to provide extra time with which to cope with a breakdown. The "protected" duration of activity i, assuming a single resource R, is then obtained as its original duration plus the duration of breakdowns that are expected to occur during its execution; i.e. $d_i' = d_i + \frac{d_i}{\mu_{ibf}(R)} \times \mu_{dt}(R)$, where $\mu_{ibf}(R)$ is the mean time between failure and $\mu_{dt}(R)$ is the mean downtime of resource R. The baseline schedule is then obtained by solving the scheduling problem with protected durations. Temporal protection will be revisited in Section 7.1.3.3.

7.1.3 Slack-based techniques

7.1.3.1 The job shop model of Leon et al.

Leon et al. (1994) describe a genetic algorithm for generating robust schedules for job shops. They define the schedule robustness of a job shop schedule S as $R(S) = r \times E[\boldsymbol{M}(S)] + (1-r)E[\boldsymbol{\delta}(S)]$, where $\boldsymbol{M}(S)$ is a random variable denoting the actual makespan of S in the presence of disruptions, r is a real-valued weight in the interval [0,1], and $\boldsymbol{\delta}(S) = \boldsymbol{M}(S) - M_o(S)$ represents the schedule delay, defined as a random variable expressing the difference between executed and pre-schedule makespan. Since $M_o(S)$ is deterministic, the expected values of $\boldsymbol{M}(S)$ and $\boldsymbol{\delta}(S)$ equate as $E[\boldsymbol{M}(S)] = E[\boldsymbol{\delta}(S)] + M_o(S)$. The authors assume a right-shift reactive policy that restarts the disrupted operations immediately after the disruption period. They demonstrate that schedule robustness $R(S)$ can be computed directly for a schedule with a single disruption. When there is more than one disruption, the authors have tested three

surrogate robustness measures. The measure $RM3(S) = M_o(S) - \frac{\sum_{i \in N_f} slack_i}{|N_f|}$, where N_f is the set of activities executing on fallible machines and $slack_i = lst_i - est_i$ denotes the *slack time* of activity i (the difference between the latest and earliest start time of the activity), is the simplest to compute. Simulation demonstrates that mean activity slack was as good a predictor of $E[\delta(S)]$ as the more sophisticated surrogates and that $RM3(S)$ performs better than the exact calculation of expected delay for the single disruption case, when only one machine in the shop is fallible.

7.1.3.2 The pairwise float model of Herroelen and Leus

7.1.3.2.1 Abstraction of resource usage

Herroelen and Leus (2002) develop mathematical programming models for the generation of stable baseline schedules in a project environment. The authors make abstraction of resource usage, assuming that a proper allocation of resources has been performed. They use the concept of *pairwise float*, $F_{ij}(S) = s_j(S) - f_i(S)$, defined as the difference between the start time of activity j and the finish time of activity i in a schedule S. The pairwise float is only defined for activities $(i, j) \in TA$, where TA denotes the transitive closure of A, meaning that $(i, j) \in TA$ if and only if i and j are connected by a path in the activity-on-the-node project network $G = (N, A)$. The authors assign a project deadline δ_n and a probability of disruption p_i to every activity i (i=1,2,…,n), with $\sum_{i=1}^{n} p_i = 1$. The dummy end node has disruption probability $p_n = 0$, while p_1 denotes the probability that the dummy start node, i.e. the entire project, starts later than initially anticipated. They use a random variable $\boldsymbol{L}_i$ to denote the disturbance length of activity i *if* it is disturbed, and a non-negative cost c_i per unit time overrun on the start time of activity i.

The authors propose to use as stability measure the expected weighted deviation in start times in the realized schedule from those in the pre-schedule. In other words, the expression they wish to minimize is $\sum_{j=1}^{n} c_j (E[\boldsymbol{s}_j] - s_j(S))$, with E the expectation operator, $s_j(S)$ the start time of activity j in the pre-schedule S, and $\boldsymbol{s}_j$ a random variable representing the actually achieved start time of activity j (after project execution). If for all arcs $(i, j) \in TA$, $MSPF_{ij}$ denotes the minimal sum of pairwise floats of all edges on any path leading from i to j, then $E[\boldsymbol{s}_j]$ can be computed as $s_j(S) + \sum_{i \in \pi^T(j)} p_i \max\{0; \boldsymbol{L}_i - MSPF_{ij}\} \mid i \text{ disturbed})$, where $\pi^T(j)$ is the set of all immediate and transitive predecessors of j. Hence, the objective can be rewritten as min $\sum_{(i,j) \in TA} c_j p_i E(\max\{0; \boldsymbol{L}_i - MSPF_{ij}\} \mid i \text{ disturbed})$. Assuming a single disruption and all $\boldsymbol{L}_i$ to be discrete, with probability mass function $g_i(.)$ which associates nonzero probability with positive values l_{ik} that correspond with the elements k in D_i, the set of disturbance scenarios for activity i, the authors solve the following linear programming model:

$$\min \quad \sum_{(i,j)\in TA} \sum_{k\in D_i} c_j p_i g_i(l_{ik}) \Delta_{ijk} \qquad (1)$$

subject to

$$s_i + d_i + F_{ij} = s_j \qquad \forall\,(i,j) \in A \qquad (2)$$

$$s_n \le \delta_n \qquad (3)$$

$$l_{ik} - MSPF_{ij} \le \Delta_{ijk} \qquad \forall\,(i,j) \in TA,\ \forall k \in D_i \qquad (4)$$

$$s_i + d_i + \lambda_{ij} + MSPF_{ij} = s_j \qquad \forall\,(i,j) \in TA \qquad (5)$$

all $\Delta_{ijk}, s_i, F_{ij}, MSPF_{ij} \ge 0$,

where Δ_{ijk} is the delay in the start time of activity j due to a disturbance according to scenario k of activity i, and λ_{ij} is the length of the path from i to j (not including i and j) for which $MSPF_{ij}$ is achieved. This linear program can be rewritten as the dual of a minimum cost network flow problem. The authors have extended the model to cope with multiple disturbances. They report on very promising computational results obtained on a set of randomly generated test instances.

7.1.3.2.2 Restricted resources

If the unrestricted resource availability assumption is dropped from the analysis, Leus and Herroelen (2001) use a so-called *resource flow network* to represent the flow of resources across the activities of the project network (the concept of a resource flow network has been presented by Naegler and Schoenherr (1989), Bowers (1995) and Artigues and Roubellat (2000)). Consider the example project and associated early start schedule shown in Figure 3. The network is shown in Figure 3(a) in activity-on-the-node format. Activities 1 and 6 are dummies. For each activity, the duration and per period requirement for a single renewable resource are shown above the corresponding node. The resource is assumed to have a constant availability of 3 units. The feasible early start schedule shown in Figure 3(b) minimizes the project duration.

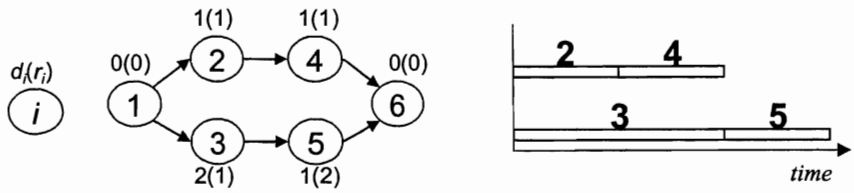


Figure 3. Example project and early start schedule

Figure 4(a) represents a possible resource flow network that uses all three available resource units. The dummy start activity passes one unit of the resource to activity 2, one unit to activity 3 and one unit to activity 5. Activity 2 passes the resource unit it received from activity 1 to activity 4. In addition to the resource unit received

from the dummy start activity, activity 5 receives the resource unit released by activity 3 upon its completion. The dummy end activity 6 receives one resource unit upon completion of activity 4 and two resource units upon completion of activity 5.

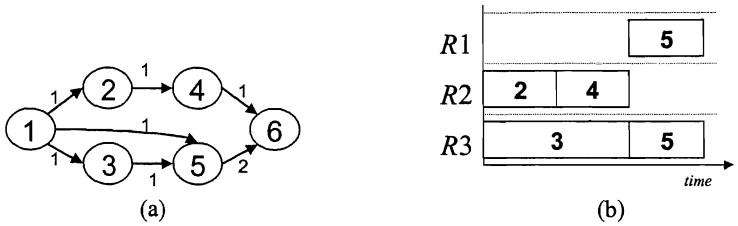


Figure 4. Resource flow network and associated early start schedule

As such the technological precedence constraints of Figure 3(a) have been augmented with the corresponding resource links yielding a resource-unconstrained network. The resulting partial order defines an early start policy (see Section 4.1) and can be captured by the set of arcs A on the condition that the resource allocation is kept fixed. Under this condition, schedule repair is trivial and the use of the pairwise float model discussed above on the resulting network allows for the generation of an optimal pre-schedule in polynomial time.

An interesting question is whether we can find a feasible resource allocation corresponding with a given feasible input schedule S such that $\sum_{(i,j)\in TA\cup F}\sum_{k\in D_i} c_j p_i g_i(l_{ik})\Delta_{ijk} \le U$, where F is the set of extra resource links. Leus (2002) has shown that this decision problem is NP-complete in the ordinary sense even when all activities have a single disruption scenario, by establishing that the parallel machine problem with weighted completion time objective (Bruno et al., 1974) can be reduced to it.

7.1.3.2.3 Robust resource allocation

Leus and Herroelen (2001) have studied the problem of generating a robust resource allocation under the assumption that a feasible baseline schedule exists and that some advance knowledge about the probability distribution of the activity durations is available. What is meant by a robust resource allocation can again be illustrated on the project of Figure 3. Figure 5(a) shows an alternative resource flow network with associated early start schedule shown in Figure 5(b). The resource allocation represented by the resource flow network of Figure 5(a) is clearly less robust than the one studied before in Figure 4(a). In Figure 4, activity 4 has one unit of slack, while now activity 4 is forced to pass on a resource unit to activity 5. The slightest start time delay or duration extension of activity 4 will have an immediate impact on the makespan of the (repaired) schedule.

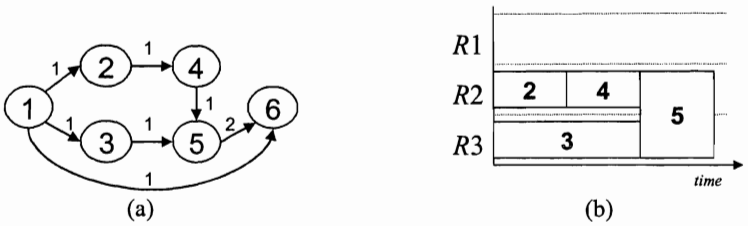


Figure 5. Resource flow network and associated early start schedule

Leus and Herroelen (2001) explore the fact that checking the feasibility of a resource allocation can easily be done using maximal flow computations in the resource flow network. As such, the search for an optimal allocation is reduced to the search for an associated resource flow network with desirable robustness characteristics. The authors propose a branch-and-bound algorithm that solves the robust resource allocation problem in exact and approximate formulations. The procedure heavily relies on constraint propagation during its search. The authors report on promising results obtained on a set of problem instances generated using the problem generator *RanGen* (Demeulemeester et al., 2001).

7.1.3.3 Time window slack

The intuition behind the temporal protection technique discussed in Section 7.1.2 is that during schedule execution, the protected durations of activities scheduled consecutively on breakable resources provide slack time that can be used in the event of resource breakdowns. If two activities i and j are scheduled consecutively on a breakable resource, and if the machine breaks down while i is executing, the extra time within its protected duration can be used to absorb the breakdown. If no breakdown occurs during the execution of activity i, then activity j can start earlier. However, it is easy to see that this can be a mere illusion. If activity j has a predecessor k on a non-breakable resource that finishes later than activity i, the temporal protection represented by the extended duration of activity i is not available for activity j as j cannot start earlier than the finish time of k.

To avoid such situations, Davenport et al. (2001) propose the *time window slack* (TWS) approach, which does not include slack as part of the activity duration, but adds a relation to the problem definition that specifies that schedules must have sufficient slack for each activity. If A_R denotes the set of activities that require resource R, the required slack for activity $i \in A_R$ is defined to be at least $\frac{\sum_{j \in A_R} d_j}{\mu_{tbf}(R)} \times \mu_{dt}(R)$, where $\mu_{tbf}(R)$ is the mean time between failure and $\mu_{dt}(R)$ is the mean down time of resource R. It should be noted that the required slack for an activity under TWS is considerably larger than the duration extension in the temporal protection technique of Section 7.1.2: the amount of slack on each activity is equal to the sum of the durations of all the expected breakdowns on the resource. The use of mean time between failure and mean downtime data make

the approach less applicable in a project setting, where most renewable resources are human beings.

A disadvantage of the temporal protection and time window slack technique is that the placement of activities on the scheduling horizon is not taken into account. Davenport et al. (2001) therefore suggest to use *focused time window slack* (FTWS) which uses the uncertainty statistics to concentrate the slack in areas of the schedule horizon that are more likely to need it to deal with a breakdown. The intuition is that the later in a schedule an activity is executed, the more likely it is to have a disruptive event occur before its execution, and therefore, the more slack is needed. The slack for an activity is computed as a function of the probability that a breakdown will occur before or during the execution of the activity and of the expected breakdown duration. The amount of slack time required for an activity executing at a particular point t on resource R should be at least equal to $\sum_{nb=1}^{M} P(N(\mu(nb),\sigma(nb)) \le t) \times \mu_{dt}(R)$, where $\mu(nb) = (nb \times \mu_{tbf}(R)) + ((nb-1) \times \mu_{dt}(R))$ denotes the expected time that the nb-th breakdown will occur, $\sigma(nb) = ((nb \times \sigma_{tbf}^2(R)) + ((nb-1) \times \sigma_{dt}^2(R)))^{\frac{1}{2}}$ and $P(N(\mu(nb),\sigma(nb)) \le t$ denotes the probability that nb breakdowns will occur before a particular time t.

Simulation results obtained on a number of randomly generated job shop instances revealed the superiority of TWS and FTWS over temporal protection both in producing schedules with low simulated tardiness and in producing schedules that better predict the level of simulated tardiness. Again, the use of breakdown probabilities make the approach less applicable in a project setting.

7.1.3.4 The float factor approach

Tavares et al. (1998) study the risk of a project as a function of the uncertainty of the duration and the cost of each activity and the adopted schedule. The adoption of an early (late) start schedule reduces (increases) the risk of an overall delay but increases (decreases) the project's discounted cost which calls for the difficult determination of an optimal compromise. The authors suggest that the start time of each activity i be set equal to $s_i(\alpha) = es_i + \alpha(ls_i - es_i)$, where es_i and ls_i denote the earliest, respectively, latest start time of activity i given project deadline δ_n, and $\alpha, 0 \le \alpha \le 1$, denotes the so-called float factor. The late start (early start) schedule is obtained with $\alpha = 1$ ($\alpha = 0$). The authors prove that the use of $s_i(\alpha)$ yields a feasible schedule.

Herroelen and Leus (2002) have adapted the float factor model allowing the float factor to vary among the project activities, in order to pursue stability in the schedule. Using the same notation as in Section 7.1.3.2 and defining, for each activity i, $\sigma^T(i)$ as the set of all its immediate and transitive successors, the authors define the quantities $\beta(i) = \sum_{l \in \pi^T(i) \cup \{i\}} p_k c_l$, the sum of weights of all arcs that precede i in the network, and $\phi(i) = \sum_{k \in \sigma^T(i) \cup \{i\}} p_k c_l$, the sum of the weights of all arcs that succeed i in the network. They then define for each activity i an *activity-dependent float factor*

$\varphi(i) = \frac{\beta(i)}{\beta(i)+\phi(i)}$. Logically $\varphi(1) = 0$ and $\varphi(n) = 1$. If $\beta(i) = \phi(i) = 0$, they choose $\varphi(i) = 0.5$ (except for $i = 1$ or n). Otherwise, $\varphi(i) = (1 + \phi(i)/\beta(i))^{-1}$, such that $\varphi(i) \leq \varphi(j)$ if $(i, j) \in TA$, so the resulting schedule will be feasible.

Results obtained on a dataset consisting of 300 instances generated using the problem generator *RanGen* (Demeulemeester et al., 2001) demonstrate that this activity-dependent float factor-based model is clearly outperformed by the model given earlier in Eqs. (1)-(5). For a single disruption the model yields an expected weighted deviation in starting times that is over 100% (119.95%) above the values obtained by the model of Eqs. (1)-(5), while this percentage still amounts to 31.45% when 2 out of every 3 activities are disrupted.

7.1.4 Idle time insertion

Mehta and Uzsoy (1998, 1999) insert additional idle time into the predictive schedule to absorb the impact of machine breakdowns. Mehta and Uzsoy (1999) consider the problem of minimizing total tardiness on a single machine with dynamic job arrival and random breakdowns. They compute an initial sequence by a heuristic and then insert additional idle times into the schedule. Mehta and Uzsoy (1998) study the problem of minimizing the maximum lateness in a job shop subject to machine breakdowns. Assuming the distributions of the time between breakdowns and the time to repair for the machines to be available, they generate a baseline schedule using the shifting bottleneck heuristic (Adams et al., 1988). They invoke earliness and lateness penalties whenever the last operation of a job ends sooner or later than planned. They use two heuristics to insert idle time to minimize expected job completion time deviations. In the "linear programming based heuristic" (LPH), the idea is to develop a schedule with *expected* durations for all the activities, and minimize the summed deviation of the pre-schedule from this 'blown up schedule'.

Herroelen and Leus (2002) have adapted the model of Mehta and Uzsoy to a project environment. As in Section 7.1.3.2.1, abstraction is made of resource usage. All definitions and symbols correspond with the referred Section. $\Omega(\gamma)$ denotes the earliest start schedule when d_i, the planned duration of activity i, is replaced by $d_i' = d_i + \gamma m p_i E[L_i]$. E denotes the expectation operator and γ measures the degree in which the expected values of disruptions are propagated throughout the network (γm is used to make the parameter independent of the number of activities n, the average probability being $1/n$). Δ_i denotes the amount by which the starting time s_i of activity i in the generously protected schedule $\Omega(\gamma)$ exceeds the pre-schedule. The authors then generate a pre-schedule according to the output of the following linear programming model:

$$\text{(LPH)} \qquad \min \quad \sum_{i \in N} c_i \Delta_i \tag{6}$$

subject to

$$s_i + d_i \leq s_j \qquad \forall (i,j) \in A \tag{7}$$

$$s_n \leq \delta_n \tag{8}$$

$$s_i(\Omega(\gamma)) = s_i + \Delta_i \qquad \forall\, i \in N \qquad (9)$$
$$\text{all } \Delta_i,\ s_i \geq 0$$

The authors demonstrate that (LPH) corresponds to the dual of a minimal cost network flow problem. Results obtained on a dataset consisting of 300 instances generated using the problem generator *RanGen* (Demeulemeester et al., 2001) demonstrate that this LPH-based model is clearly outperformed by the model given earlier in Eqs. (1)-(5). For a single disruption the model yields an expected weighted deviation in starting times that is 158.94% above the values obtained by the model of Eqs. (1)-(5), while for 20 disturbances this percentage still amounts 34.4%.

7.1.5 Buffer insertion (critical chain)

Critical Chain Scheduling/Buffer Management (*CC/BM*) – the direct application of the Theory of Constraints (*TOC*) to project management (Goldratt, 1997) – has received a lot of attention in the project management literature. The fundamentals of *CC/BM* are summarized in Table 2 (Herroelen et al., 2002).

Table 2. *CC/BM* fundamentals
50% probability activity duration estimates
No activity due dates
No project milestones
No multi-tasking
Scheduling objectives = minimize makespan; minimize WIP
Determine a precedence and resource feasible *baseline schedule*
Identify the critical chain
Aggregate uncertainty allowances into buffers
Keep the baseline schedule and the critical chain fixed during project execution
Determine an early start based unbuffered *projected schedule* and report early completions (apply the roadrunner mentality)
Use the buffers as a proactive warning mechanism during schedule execution

CC/BM builds a *baseline schedule* using activity duration estimates based on a 50% confidence level. Activity due dates and project milestones are eliminated and multi-tasking is to be avoided. In order to minimize work-in-progress (WIP), a precedence feasible schedule is constructed by scheduling activities at their latest start times based on critical path calculations. If resource conflicts occur, they are resolved by moving activities earlier in time. The *critical chain* is then defined as that chain of precedence and resource dependent activities which determines the overall duration of a project. If there is more than one critical chain, just select one. The safety associated with the critical chain activities that was cut away by selecting aggressive duration estimates is shifted to the end of the critical chain in the form of a *project buffer* (PB). This project buffer should protect the project due date promised to the customer from variability in the critical chain activities. *Feeding buffers* (FB) are inserted whenever a non-critical chain activity joins the critical chain. Their aim is to protect the critical chain from disruptions on the activities feeding it, and to allow critical chain activities to start early in case things go well. Although more detailed methods can be used for sizing the

buffers (Newbold, 1998; Product Development Institute, 1999), the default procedure is to use the 50% *buffer sizing rule*, i.e., to use a project buffer of half the project duration and to set the size of a feeding buffer to half the duration of the longest non-critical chain path leading into it. *Resource buffers* (RB), usually in the form of an advance warning, are placed whenever a resource has to perform an activity on the critical chain, and the previous critical chain activity is done by a different resource.

The *CC/BM* baseline schedule for the project of Figure 1 would allow for the identification of 16 critical chains. ProChain®, one of the best-known software packages that can be used for implementing *CC/BM*, would select the critical chain 4-7-8-9. Using the 50% buffer rule, the buffered baseline schedule of Figure 6, generated by ProChain®, would have a two-period feeding buffer to protect the project buffer (!) from variation in activity 2, a three-period feeding buffer to protect critical chain activity 9 from variation in the path 3-6, and a one-period feeding buffer to protect critical chain activity 8 from variation in activity 5. The reader will observe the rather surprising phenomenon that the critical chain is broken, i.e., it is no longer a contiguous chain that determines the project duration since it contains gaps. Moreover, there is no apparent reason why the "critical chain" would have a one-period gap between activity 7 and 8 and activities 8 and 9. The feeding buffer in front of activity 8 is redundant, given the preceding two-period gap in the schedule. The software inserts a resource buffer in front of activity 7 to give a warning signal to the extra resource unit needed for the execution of critical chain activity 7. In this way a warning signal is given to a resource used for the execution of a critical chain activity that no longer determines the project duration.

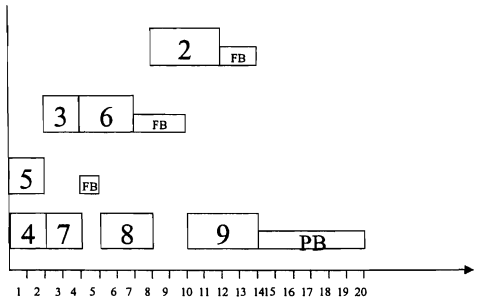


Figure 6. Buffered baseline schedule for the project of Figure 1

Herroelen and Leus (2001) have validated the working principles of *CC/BM* through a full factorial computational experiment using the well-known 110 Patterson test problems (Patterson, 1984). Contrary to *CC/BM* belief, they reach the conclusion that (a) updating the baseline schedule and the critical chain at each decision point provides the best intermediate estimates of the final project duration and yields the smallest final project duration, (b) using a clever project scheduling mechanism such as branch-and-bound has a beneficiary effect on the final makespan, the percentage deviation from the optimal final makespan obtainable if information would be perfect, and the work-in-progress, (c) using the 50% rule for buffer sizing may lead to a serious overestimation of the project buffer size, (d) the beneficiary effect of computing the

buffer sizes using the root-square-error method increases with problem size, (e) keeping the critical chain activities in series is harmful to the final project makespan, and (f) recomputing the baseline schedule at each decision point has a strong beneficiary impact on the final project duration.

7.2 *Quality robust schedules*

7.2.1 Min-max regret techniques

Min-max regret techniques view schedule robustness as the determination of the schedule with the best worst-case performance compared to the corresponding optimal solution over all potential realizations of job processing times. The approach assumes a set of discrete processing time scenarios each of which specifies the processing time of each job (Kouvelis and Yu, 1997). Using scenarios to structure variability allows the decision maker to describe the relationship between uncertain factors in the scheduling environment and corresponding job processing times in the most appropriate manner based on internal knowledge and experience. In this manner correlation among major factors that affect job processing times can be easily accommodated. The generation of processing time scenarios also provides insight into the nature of the scheduling environment by requiring the decision maker to identify events that have occurred, or almost certainly will occur, but whose consequences have yet to unfold, and to formalize the perceived connections among events and forces that drive the scheduling environment (Daniels and Kouvelis, 1995).

Daniels and Kouvelis (1995) study the single machine problem under the total flow time objective, while Kouvelis et al. (2000) focus on the two-machine flow shop environment. Their objective is to *minimize the maximum possible regret* associated with a schedule. For a given schedule and a set of processing times for the single machine problem, the regret is measured as the absolute difference between the total flow time of the schedule for that scenario and the flow time obtained using the (optimal) shortest processing time rule. For the two-machine flow shop problem the deviation is computed between the makespan of the schedule for a scenario and the makespan of the (optimal) Johnson schedule for that scenario. The authors develop branch-and-bound algorithms and heuristics for determining robust schedules.

7.2.2 Quality-robust evaluation functions

The min-max regret approach requires advance knowledge of all possible execution scenarios and advance knowledge of the optimal solution for each scenario. Sevaux and Sörensen (2002) study the single machine scheduling problem with ready times under the objective of minimizing the weighted number of late jobs. They rely on a genetic algorithm for generating quality robust schedules, i.e., schedules whose quality does not change when the input data (i.e., the ready times) change. The authors use a robust evaluation function $f_r(x) = \frac{1}{m}\sum_{i=1}^{m} w_i f(x+\delta_i)$. This evaluation function adds some noise δ_i to the current solution x before an evaluation; the final evaluation is the average over m disrupted solutions. A disruption amounts to a modification of the ready time (by

adding a random value δ_i between 0 and δ_{max}) of between 0 and 20% of the total number of jobs. The authors fix the number of evaluations m at 10 and conclude that the value of the objective function remains high when small variations in some ready times occur.

7.2.3 β-robust schedules

Consistent with the min-max regret philosophy, Daniels and Carrillo (1997) opt for a scheduling approach that considers both average system performance and performance variability in determining the optimal schedule. Focusing on a single machine environment and a set of activity processing time scenarios, their scheduling objective is to determine a β-robust schedule, i.e., the schedule with the maximum likelihood of achieving flow time performance no greater than a particular target level. Having established NP-hardness of the problem, the authors offer a branch-and-bound procedure and a heuristic. They also extend the analysis to those situations where a single resource, available in limited supply, can be applied to individual jobs to linearly decrease the associated processing time variance. Computational experience indicated that β-robust schedules provide effective hedges against processing time uncertainty while maintaining near-optimal performance with respect to expected flow time.

7.3 *Multiple schedules* (*contingent scheduling*)

The contingent scheduling approach is based on the idea to generate multiple baseline schedules (or baseline schedule fragments) which optimally respond to anticipated disruptive events. Responding to unexpected but anticipated events during schedule execution is then simply done by switching to the schedule (fragment) that corresponds to the events that have occurred.

7.3.1 Just-in-Case Scheduling

Bresina et al. (1994) have developed the technique of just-in-case scheduling in the domain of telescope observation scheduling. The technique is based on the identification of high probability schedule breaks and the generation of an alternative schedule for each break, just in case the break occurs during execution. In overview, the algorithm accepts a schedule as input and using a model of how durations can vary, the temporal uncertainty at each step in the schedule is estimated. The most probable break during this uncertainty is determined and the break point is split into two cases: one in which the schedule breaks and one in which it does not. The scheduler is then invoked on a new scheduling sub-problem to produce an alternative schedule for the break case. This alternative schedule is integrated with the initial schedule.

The authors report computational experience on real telescope scheduling data, involving one machine and fewer than 20 schedule breaks. As already observed by Davenport and Beck (2002), this approach runs into combinatorial problems when more than one resource is involved.

7.3.2 Group sequences

Billaut and Roubellat (1996a) suggest to generate for every resource a so-called *group sequence*, i.e. a totally or partially ordered set of groups of operations, and to consider all the schedules obtained by an arbitrary choice of the ordering of the operations inside each group. Maugière et al. (2002) and Aloulou et al. (2002) explore this sequence flexibility idea in the context of single machine scheduling.

The gist of the approach can be sketched using the 4 job-2 machine example borrowed from Billaut and Roubellat (1996a). The four jobs are subject to ready times $\rho_1 = 1, \rho_2 = \rho_3 = \rho_4 = 0$ and due dates $\delta_2 = 4, \delta_1 = \delta_3 = \delta_4 = 5$. Additional data are shown in Table 3. The notation (i,j) refers to operation j of job i. Consider the following group sequence:

Resource 1: group 1: {(1,1),(2,1)} group 2: {(3,2),(4,2)}
Resource 2: group 1: {(3,1),(4,1)} group 2: {(1,2),(2,2)}

Table 4 enumerates the 16 schedules that can be generated from this group sequence by choosing an arbitrary processing order for the operations inside each group ($a \prec b$ means a strictly precedes b). All sixteen schedules are feasible. In this way the decision maker is not only provided with one feasible schedule but several ones. The hope is that during the real-time execution of the schedule, it becomes possible to switch from one solution to the other in the presence of a disruption without any loss of performance.

Table 3. Data for the 4 job-2 resource problem

(job/operation)	(1,1)	(1,2)	(2,1)	(2,2)	(3,1)	(3,2)	(4,1)	(4,2)
machine	1	2	1	2	2	1	2	1
processing time	1	1	1	1	1	1	1	1

Table 4. Set of schedules (Billaut and Roubellat (1996a)

Resource 1	Resource 2
(1,1) ≺ (2,1) ≺ (3,2) ≺ (4,2)	(3,1) ≺ (4,1) ≺ (1,2) ≺ (2,2)
(2,1) ≺ (1,1) ≺ (3,2) ≺ (4,2)	(3,1) ≺ (4,1) ≺ (1,2) ≺ (2,2)
(1,1) ≺ (2,1) ≺ (4,2) ≺ (3,2)	(3,1) ≺ (4,1) ≺ (1,2) ≺ (2,2)
(2,1) ≺ (1,1) ≺ (4,2) ≺ (3,2)	(3,1) ≺ (4,1) ≺ (1,2) ≺ (2,2)
(1,1) ≺ (2,1) ≺ (3,2) ≺ (4,2)	(4,1) ≺ (3,1) ≺ (1,2) ≺ (2,2)
(2,1) ≺ (1,1) ≺ (3,2) ≺ (4,2)	(4,1) ≺ (3,1) ≺ (1,2) ≺ (2,2)
(1,1) ≺ (2,1) ≺ (4,2) ≺ (3,2)	(4,1) ≺ (3,1) ≺ (1,2) ≺ (2,2)
(2,1) ≺ (1,1) ≺ (4,2) ≺ (3,2)	(4,1) ≺ (3,1) ≺ (1,2) ≺ (2,2)
(1,1) ≺ (2,1) ≺ (3,2) ≺ (4,2)	(3,1) ≺ (4,1) ≺ (2,2) ≺ (1,2)
(2,1) ≺ (1,1) ≺ (3,2) ≺ (4,2)	(3,1) ≺ (4,1) ≺ (2,2) ≺ (1,2)
(1,1) ≺ (2,1) ≺ (4,2) ≺ (3,2)	(3,1) ≺ (4,1) ≺ (2,2) ≺ (1,2)
(2,1) ≺ (1,1) ≺ (4,2) ≺ (3,2)	(3,1) ≺ (4,1) ≺ (2,2) ≺ (1,2)
(1,1) ≺ (2,1) ≺ (3,2) ≺ (4,2)	(4,1) ≺ (3,1) ≺ (2,2) ≺ (1,2)
(2,1) ≺ (1,1) ≺ (3,2) ≺ (4,2)	(4,1) ≺ (3,1) ≺ (2,2) ≺ (1,2)
(1,1) ≺ (2,1) ≺ (4,2) ≺ (3,2)	(4,1) ≺ (3,1) ≺ (2,2) ≺ (1,2)
(2,1) ≺ (1,1) ≺ (4,2) ≺ (3,2)	(4,1) ≺ (3,1) ≺ (2,2) ≺ (1,2)

Billaut and Roubellat (1996ab) extend the group sequence concept to the multiple renewable resource case by adding the condition that the operations in a group should use the same amount of a resource type, and the operations in a group are assigned to the same subset of units of the resource type. Briand et al. (2002) extend the

methodology used by Billaut and Roubellat (1996b) to the case of multi-mode scheduling with minimal and maximal time-lags.

Artigues et al. (1999) study multi-mode project scheduling problems where the projects have a release date and a due date. They propose a generation procedure for finding group sequences based on a new priority rule. They also propose and test an efficient local search procedure to improve the feasibility of a group sequence. The procedures are integrated in a commercial real-time scheduling package (ORDO®).

8. Sensitivity analysis

A number of recent research efforts focus on the sensitivity analysis of machine scheduling problems (Hall and Posner, 2000ab). Sensitivity analysis addresses "What if...?" types of questions that arise from parameter changes. The authors study polynomially solvable and intractable machine scheduling problems and try to provide answers to a number of fundamental questions such as (a) what are the limits to the change of a parameter such that the solution remains optimal?, (b) given a specific change of a parameter, what is the new optimal cost?, (c) given a specific change of a parameter, what is a new optimal solution?, (d) when does a baseline schedule remain optimal?, (e) when does the objective function value remain optimal?, (f) what types of sensitivity analysis are useful to evaluate the robustness of optimal solutions?, (g) what types of sensitivity analysis can be performed without using the full details of the solution?, etc. An interesting area of future research is to pose and answer similar questions in a project scheduling setting. An additional interesting and as yet unexplored research topic is to determine what parameter changes are allowed to guarantee full rescheduling optimality by means of a 'simple' repair action (e.g. right shift).

Burns et al. (1997) use sensitivity analysis in combination with schedulability analysis to establish the maximum fault frequency that a single processor system on which a finite number of tasks must be repeatedly executed can tolerate. Each task has a minimum inter-arrival time, a worst-case execution time and a deadline. The problem is to define a policy that guarantees that each instance of each task will finish by its deadline. The authors use sensitivity analysis to find the minimum value of the minimum fault inter-arrival time such that all tasks meet their deadlines. The authors do not provide computational results.

Penz et al. (2001) determine the *sensitivity guarantee* of off-line scheduling algorithms for single and parallel machine scheduling problems where the actual duration of a task i is equal to $(1+\varepsilon_i)d_i$, with $\varepsilon_i \in]-1,+\infty[$ representing the percentage of confidence we have on the corresponding estimated duration. Values $1+\varepsilon_i$ are the components of the perturbation vector $\vec{\varepsilon}$. The sensitivity guarantee of an off-line algorithm *ALG* is a function $s_{ALG}(\varepsilon)$ such that for any off-line instance $\Im$ and any ε-perturbation $\vec{\varepsilon}$, $s_{ALG}(\varepsilon)$ is the smallest real value verifying $\rho^{\vec{\varepsilon}}_{ALG}(\Im) \le s_{ALG}(\varepsilon)\rho_{ALG}(\Im)$. In this expression, $\rho_{ALG}(\Im) = f_{ALG}(\Im)/f_{OPT}(\Im)$ denotes the *theoretical* or *off-line* performance ratio of algorithm *ALG* for which $f_{ALG}(\Im)$ denotes the objective value achieved by algorithm *ALG* on $\Im$ and $f_{OPT}(\Im)$ denotes the optimal objective value for the instance. $\rho^{\vec{\varepsilon}}_{ALG}(\Im) = f^{\vec{\varepsilon}}_{ALG}(\Im)/f^{\vec{\varepsilon}}_{OPT}(\Im)$ denotes the *effective* performance ratio, i.e. obtained after execution. The numerator and denominator in the right-hand side of the

expression represent the objective value of the *ALG* schedule for $\mathfrak{I}$, applied to $\mathfrak{I}$ perturbed by $\vec{\varepsilon}$, and the optimal value *ex post*, with perfect knowledge, respectively.

9. Conclusions and suggestions for further research

The majority of research efforts in project scheduling assume complete information about the scheduling problem to be solved and assume a static deterministic environment. Basically the research efforts aim at the generation of feasible baseline schedules that 'satisfice' or optimize single or multiple objective functions. The literature on project scheduling under risk and uncertainty is rather sparse. In this paper we offer a review of the major approaches to deal with scheduling risk and uncertainty, many of which have been mainly or solely studied in a machine scheduling environment.

The methodologies for *stochastic project scheduling* basically view the project scheduling problem as a multi-stage decision process. Scheduling policies are used that define which activities are to be started at random decision points through time, based on the observed past and the a priori knowledge about the processing time distributions. As such they share the disadvantage that they do not explicitly generate a pre-schedule that can be used as the baseline plan for making advance commitments to both subcontractors and customers. The dynamic programming approaches developed to tackle the stochastic multi-mode problem determine the resource allocation vectors for the project activities in order to minimize total expected cost and rely on the assumption that the uncertainty resides in the work content of the activities and not in their duration.

The temporal analysis of GERT *networks* has been widely studied. The problem, however, are the heavy computational requirements forcing the use of simulation techniques. GERT *network scheduling* has only received recent attention. Virtually all the models assume the resources to be machines. Again the heavy computational burden prohibits the practical use of analytical approaches.

The *fuzzy project scheduling* approach rejects the use of probability distributions for the activity durations but relies on membership functions that may be as difficult to generate. As such uncertainty is captured by the notion of "belonging" rather than in terms of "frequency" of occurrence. The literature is still in its burn-in phase.

Research in *proactive* (*robust*) *scheduling* has widely prospered in the field of machine scheduling. Redundancy-based techniques have already found their way to the field of project scheduling. The buffer insertion approach, the fundamental ingredient of Goldratt's critical chain methodology, is gaining increasing popularity among project management practitioners. While this methodology has acted as an important eye-opener, its pitfalls, mainly due to severe oversimplifications, have been revealed recently. The generation of robust multi-resource baseline schedules in combination with efficient and effective *reactive schedule repair* mechanisms constitutes a viable area of future research. Whereas numerous reactive scheduling mechanisms have been developed and tested in real-time machine scheduling environments, the field is in need for further research aimed at their implementation and validation in a project scheduling environment.

Research on *sensitivity analysis* has just emerged in the area of machine scheduling. Efforts to seek answers to the various types of "what if ..." questions in a

project setting still need to be initiated, and would offer useful information to project management.

References

Adams, J., E. Balas and D. Zawack, 1988, The Shifting Bottleneck Procedure for Job Shop Scheduling, *Management Science*, 34, 391-401.

Akturk, M.S. and E. Gorgulu, 1999, Match-up Scheduling Under a Machine Breakdown, *European Journal of Operational Research*, 112, 81-97.

Alagoz, O. and M. Azizoglu, 2001, Rescheduling Under Machine Eligibility Constraints, Paper presented at the INFORMS 2001 Annual Meeting, Miami Beach, November 4-7.

Aloulou, M.A., M.-C. Portmann and A. Vignier, 2002, Predictive-Reactive Scheduling for the Single Machine Problem, Paper presented at the 8th Workshop on Project Management and Scheduling, Valencia, 3-5 April.

Artigues, C. and F. Roubellat, 2000, A Polynomial Activity Insertion Algorithm in a Multi-Resource Schedule with Cumulative Constraints and Multiple Modes, *European Journal of Operational Research*, 127, 297-316.

Artigues, C., F. Roubellat and J.-C. Billaut, 1999, Characterization of a Set of Schedules in a Resource-Constrained Multi-Project Scheduling Problem with Multiple Modes, *International Journal of Industrial Engineering – Theory Applications and Practice*, 6(2), 112-122.

Bean, J.C., J.R. Birge, J. Mittenthal and C.E. Noon, 1991, Match-up Scheduling with Multiple Resources, Release Dates and Disruptions, *Operations Research*, 39(3), 470-483.

Bell, C.E. and K. Park, 1990, Solving Resource-Constrained Project Scheduling Problems by A* Search, *Naval Research Logistics*, 37, 61-84.

Billaut, J.C. and F. Roubellat, 1996a, A New Method for Workshop Real Time Scheduling, *International Journal of Production Research*, 34(6), 1555-1579.

Billaut, J.C. and F. Roubellat, 1996b, Characterization of a Set of Schedules in a Multiple Resource Context, *Journal of Decision Systems*, 5(1-2), 95-109.

Bowers, J.A., 1995, Criticality in Resource Constrained Networks, *Journal of the Operational Research Society*, 46, 80-91.

Bresina, J., M. Drummond and K. Swanson, 1994, Managing Action Duration Uncertainty with Just-In-Case Scheduling, Proceedings of the 1994 AAAI Spring Symposium on Decision Theoretic Planning, AAAI Press, Stanford, CA.

Briand, C., E. Despontin and F. Roubellat, 2002, Scheduling with Time Lags and Preferences: A Heuristic, Paper presented at the 8th Workshop on Project Management and Scheduling, Valencia, 3-5 April.

Bruno, J., E.G.Jr. Coffman and R. Sethi, 1974, Scheduling Independent Tasks to Reduce Mean Finishing Time, *Communications of the ACM*, 17(7), 382-387.

Burns, A., S. Punnekkat, B. Littlewood and D.R. Wright, 1997, Probabilistic Guarantees for Fault-Tolerant Real-Time Systems, Technical Report DeVa TR N0 44, Design for Validation, Esprit Long Term Research Project, N0 20072.

Czyzak, P. and A. Jaskievicz, 1996, Metaheuristic Technique for Solving Multiobjective Investment Planning Problem, *Control and Cybernetics*, 25, 177-187.

Daniels, R.L. and J.E. Carrillo, 1997, β-Robust Scheduling for Single-Machine Systems with Uncertain Processing Times, *IIE Transactions*, 29, 977-985.

Daniels, R.L. and P. Kouvelis, 1995, Robust Scheduling to Hedge Against Processing Time Uncertainty in Single-stage Production, *Management Science*, 41(2), 363-376.

Davenport, A.J. and J.C. Beck, 2002, A Survey of Techniques for Scheduling with Uncertainty, unpublished manuscript, available at http://www.eil.utoronto.ca/profiles/chris/gz/uncertainty-survey.ps.

Davenport, A.J., C. Gefflot and J.C. Beck, 2001, Slack-based Techniques for Robust Schedules, Paper presented at the Constraints and Uncertainty Workshop, Seventh International Conference on Principles and Practice of Constraint Programming, November 26 – December 1, Paphos, Cyprus.

Demeulemeester, E., Vanhoucke, M. and W. Herroelen, 2001, A Random Generator for Activity-on-the-node Networks, *Journal of Scheduling*, to appear.

Demeulemeester, E.L. and W.S. Herroelen, 2002, *Project Scheduling – A Research Handbook*, Kluwer Academic Publishers, Boston.

Dorn, J., R. Kerr and G. Thalhammer, 1995, Reactive Scheduling: Improving Robustness of Schedules and Restricting the Effects of Shop Floor Disturbances by Fuzzy Reasoning, *International Journal of Human-Computer Studies*, 42, 687-704.

Elmaghraby, S.E.E., 2000, Optimal Resource Allocation and Budget Estimation in Multimodal Activity Networks, Research Paper, North Carolina State University at Raleigh, North Carolina, U.S.A.

El Sakkout, H., T. Richards and M. Wallace, 1998, Minimal Perturbance in Dynamic Scheduling, Proceedings of the Thirteenth European Conference on Artificial Intelligence (ECAI-98).

Fernandez, A.A., 1995, The Optimal Solution to the Resource-Constrained Project Scheduling Problem with Stochastic Task Durations, Unpublished Doctoral Dissertation, University of Central Florida.

Fernandez, A.A., R.L. Armacost and J. Pet-Edwards, 1996, The Role of the Non-Anticipativity Constraint in Commercial Software for Stochastic Project Scheduling, *Computers and Industrial Engineering*, 31, 233-236.

Fernandez, A.A., R.L. Armacost and J. Pet-Edwards, 1998, Understanding Simulation Solutions to Resource-Constrained Project Scheduling Problems with Stochastic Task Durations, *Engineering Management Journal*, 10, 5-13.

Fortemps, P., 1997, *Fuzzy Sets for Modelling and Handling Imprecision and Flexibility*, PhD Dissertation, Faculté Polytechnique de Mons, Belgium.

Gao, H., 1995, Building Robust Schedules Using Temporal Protection – An Empirical Study of Constraint Based Scheduling Under Machine Failure Uncertainty, Master's Thesis, Department of Industrial Engineering, University of Toronto.

Ghosh, S., 1996, Guaranteeing Fault Tolerance through Scheduling in Real-Time Systems, Ph.D. Thesis, University of Pittsburgh.

Ghosh, S., R. Melhem and D. Mossé, 1995, Enhancing Real-Time Schedules to Tolerate Transient Faults, Real-Time Systems Symposium.

Gillies, D.W. and J.W.S. Liu, 1995, Scheduling Tasks with AND/OR Precedence Constraints, *SIAM Journal on Computing*, 24, 797-810.

Glover, F. and M. Laguna, 1997, *Tabu Search*, Kluwer Academic Publishers, Boston.

Goldratt, E., 1997, *Critical Chain*, The North River Press.

Golenko-Ginsburg, D. and A. Gonik, 1997, Stochastic Network Project Scheduling with Non-Consumable Limited Resources, *International Journal of Production Economics*, 48, 29-37.

Graham, R.L., 1966, Bounds on Multiprocessing Timing Anomalies, *Bell System Technical Journal*, 45, 1563-1581.

Gutjahr, W.J., C. Strauss and E. Wagner, 2000, A Stochastic Branch-and-Bound Approach to Activity Crashing in Project Management, *INFORMS Journal on Computing*, 12(2), 125-135.

Hall, N. and M. Posner, 2000a, Sensitivity Analysis for Intractable Scheduling Problems, Research paper, The Ohio State University.

Hall, N. and M. Posner, 2000b, Sensitivity Analysis for Efficiently Solvable Scheduling Problems, Research paper, The Ohio State University.

Hapke, M., A. Jaskievicz and R. Slowinski, 1994, Fuzzy Project Scheduling System for Software Development, *Fuzzy Sets and Systems*, 21, 101-117.

Hapke, M., A. Jaskievicz and R. Slowinski, 1999, Fuzzy Multi-Mode Resource-Constrained Project Scheduling with Multiple Objectives, Chapter 16 in Weglarz, J. (ed.), *Project Scheduling – Recent Models, Algorithms and Applications*, Kluwer Academic Publishers, 355-382.

Hapke, M. and R. Slowinski, 1996, Fuzzy Priority Heuristics for Project Scheduling, *Fuzzy Sets and Systems*, 83, 291-299.

Hapke, M. and R. Slowinski, 2000, Fuzzy Set Approach to Multi-Objective and Multi-Mode Project Scheduling Under Uncertainty, Chapter 9 in Slowinski, R. and M. Hapke (eds.), 2000, *Scheduling Under Fuzziness*, Physica-Verlag, Heidelberg, 197-221.

Herroelen, W. and R. Leus, 2001, On the Merits and Pitfalls of Critical Chain Scheduling, *Journal of Operations Management*, 19, 559-577.

Herroelen, W. and R. Leus, 2002, On the Construction of Stable Project Baseline Schedules, Research Report 0220, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.

Herroelen, W., R. Leus and E. Demeulemeester, 2002, Critical Chain Project Scheduling: Do Not Oversimplify, *Project Management Journal*, to appear.

Igelmund, G. and F.J. Radermacher, 1983a, Preselective Strategies for the Optimization of Stochastic Project Networks under Resource Constraints, *Networks*, 13, 1-28.

Igelmund, G. and F.J. Radermacher, 1983b, Algorithmic Approaches to Preselective Strategies for Stochastic Scheduling Problems, *Networks*, 13, 29-48.

Jensen, M.T., 2001, Improving Robustness and Flexibility of Tardiness and Total Flow-time Job Shops Using Robustness Measures, *Applied Soft Computing*, 1, 35-52.

Jørgenson, T., 1999, *Project Scheduling – A Stochastic Dynamic Decision Problem*, Doctoral Dissertation, Norwegian University of Science and Technology, Trondheim, Norway.

Kolisch, R. and A. Sprecher, 1996, PSPLIB – A Project Scheduling Library, *European Journal of Operational Research*, 96, 205-216.

Kouvelis, P., R.L. Daniels and G. Vairaktarakis, 2000, Robust Scheduling of a Two-Machine Flow Shop with Uncertain Processing Times, *IIE Transactions*, 32, 421-432.

Kouvelis, P. and G. Yu, 1997, *Robust Discrete Optimization and Its Applications,* Kluwer Academic Publishers, Boston.

Laguna, M., P. Lino, A. Pérez, S. Quintanilla and V. Valls, 2000, Minimizing Weighted Tardiness of Jobs with Stochastic Interruptions in Parallel Machines, *European Journal of Operational Research*, 127, 444-457.

Leon, V.J., S.D. Wu and R.H. Storer, 1994, Robustness Measures and Robust Scheduling for Job Shops, *IIE Transactions*, 26(5), 32-43.

Le Pape, C., 1991, Constraint Propagation in Planning and Scheduling, CIFE Technical Report, Robotics Laboratory, Department of Computer Science, Stanford University.

Leus, R., 2002, unpublished note.

Leus, R. and W. Herroelen, 2001, Models for Robust Resource Allocation in Project Scheduling, Research Report 0128, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.

Lootsma, F.A., 1997, *Fuzzy Logic for Planning and Decision Making*, Kluwer Academic Publishers, Dordrecht.

Mauguière, P., J.-C. Billaut and C. Artigues, 2002, Grouping Jobs on a Single Machine with Heads and Tails to Represent a Family of Dominant Schedules, Paper presented at the 8th Workshop on Project Management and Scheduling, Valencia, 3-5 April.

Mehta, S.V. and R.M. Uzsoy, 1998, Predictable Scheduling of a Job Shop Subject to Breakdowns, *IEEE Transactions on Robotics and Automation*, 14(3), 365-378.

Mehta, S.V. and R. Uzsoy, 1999, Predictive Scheduling of a Single Machine Subject to Breakdowns, *International Journal of Computer Integrated Manufacturing*, 12, 1, 15-38.

Möhring, R.H., F.J. Radermacher and G. Weiss, 1984, Stochastic Scheduling Problems I – General Strategies, *ZOR – Zeitschrift für Operations Research*, 28, 193-260.

Möhring, R.H., F.J. Radermacher and G. Weiss, 1985, Stochastic Scheduling Problems II – Set Strategies, *ZOR – Zeitschrift für Operations Research*, 29, 65-104.

Möhring, R.H., M. Skutella and F. Stork, 2000, Scheduling with AND/OR Precedence Constraints, Technical Report 689/2000, Technische Universität Berlin, Department of Mathematics, Germany.

Möhring, R.H. and F. Stork, 2000, Linear Preselective Policies for Stochastic Project Scheduling, *Mathematical Methods of Operations Research*, 52, 501-515.

Naegler, G. and S. Schoenherr, S., 1989, Resource Allocation in a Network Model – The Leinet System, in Slowinski, R. and J. Weglarz (eds.), *Advances in Project Scheduling*, Elsevier.

Neumann, K., 1984, Recent Developments in Stochastic Activity Networks, *INFOR*, 22(3), 219-248.

Neumann, K., 1990, *Stochastic Project Networks*, Lecture Notes in Economics and Mathematical Systems, Vol. 344, Springer, Berlin.

Neumann, K., 1999, Scheduling of Projects with Stochastic Evolution Structure, Chapter 14 in Weglarz, J. (ed.), *Project Scheduling – Recent Models, Algorithms and Applications*, Kluwer Academic publishers, Boston, 309-332.

Neumann, K. and U. Steinhardt, 1979, *GERT Networks and the Time-Oriented Evaluation of Projects*, Lecture Notes in Economics and Mathematical Systems, Vol. 172, Springer, Berlin.

Newbold, R.C., 1998, Project Management in the Fast Lane – Applying the Theory of Constraints, The St. Lucie Press, Cambridge.

Özdamar, L. and E. Alanya, 2000, Uncertainty Modelling in Software Development Projects (with case study), *Annals of Operations Research*, 102, 157-178.

Patterson, J.H., 1984, A Comparison of Exact Procedures for Solving the Multiple Constrained Resource Project Scheduling Problem, *Management Science*, 30, 854-867.

Penz, B., C. Rapine and D. Trystram, 2001, Sensitivity Analysis of Scheduling Algorithms, *European Journal of Operational Research*, 134, 606-615.

Pet-Edwards, J., 1996, A Simulation and Genetic Algorithm Approach to Stochastic Resource-Constrained Project Scheduling, Southcon Conference Record 1996, IEEE, Pascataway, NJ, 333-338.

Pet-Edwards, J., B. Selim, R.L. Armacost and A. Fernandez, 1998, Minimizing Risk in Stochastic Resource-Constrained Project Scheduling, Paper presented at the INFORMS Fall Meeting, Seattle, October 25-28.

Pritsker, A.A.B., 1977, *Modeling and Analysis Using Q-GERT Networks*, John Wiley & Sons, New York.

Pritsker, A.A.B., 1986, *Introduction to Simulation and SLAM* II, John Wiley & Sons, New York.

Pritsker, A.A.B. and C.E. Sigal, 1983, *Management Decision Making – A Network Simulation Approach*, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Product Development Institute, 1999, Tutorial: Goldratt's Critical Chain Method, A One-Project Solution, http://www.pdinstitute.com/tutorialintro.html.

Radermacher, F.J., 1985, Scheduling of Project Networks, *Annals of Operations Research*, 4, 227-252.

Rommelfanger, H., 1990, FULPAL: An Interactive Method for Solving (Multiobjective) Fuzzy Linear Programming Problems, Section 5 in, Slowinski, R. and J. Teghem (eds.), *Stochastic Versus Fuzzy Approaches to Multiobjective Mathematical Programming Under Uncertainty*, Kluwer Academic Publishers, Dordrecht, 279-299.

Sabuncuoglu, I. and M. Bayiz, 2000, Analysis of Reactive Scheduling Problems in a Job Shop Environment, *European Journal of Operational Research*, 126, 567-586.

Sadeh, N., S. Otsuka and R. Schelback, 1993, Predictive and Reactive Scheduling with the MicroBoss Production Scheduling and Control System, Proceedings of the IJCAI-93 Workshop on Knowledge-Based Production Planning, Scheduling and Control, 293-306.

Selke, E. and R.M. Kerr, 1994, Knowledge-based Reactive Scheduling, *Production Planning and Control*, 5(2), 124-145.

Sevaux, M. and K. Sörensen, 2002, A Genetic Algorithm for Robust Schedules, Paper presented at the 8th International Workshop on Project Management and Scheduling, Valencia, April 3-5.

Slowinski, R. and M. Hapke (eds.), 2000, *Scheduling Under Fuzziness*, Physica-Verlag, Heidelberg.

Smith, S.S., 1994, Reactive Scheduling Systems, in Brown, D. E. and W. T. Scherer, *Intelligent Scheduling Systems*, Kluwer.

Sörensen, K., 2001, Tabu Searching for Robust Solutions, Proceedings of the 4th Metaheuristics International Conference, July 16-20, Porto, Portugal, 707-712.

Stork, F., 2000, Branch-and-Bound Algorithms for Stochastic Resource-Constrained Project Scheduling, Research Report No. 702/2000, Technische Universität Berlin.

Stork, F., 2001, Stochastic Resource-Constrained Project Scheduling, Ph.D. Thesis, Technische Universität Berlin.

Tavares, L.V., J.A.A. Ferreira and J.S. Coelho, 1998, On the Optimal Management of Project Risk, *European Journal of Operational Research*, 107, 451-469.

Tsai, Y.W. and D.D. Gemmil, 1996, Using a Simulated Annealing Algorithm to Schedule Activities of Resource-Constrained Projects, Working Paper No. 96-124, Iowa State University.

Tsai, Y.W. and D.D. Gemmil, 1998, Using Tabu Search to Schedule Activities of Stochastic Resource-Constrained Projects, *European Journal of Operational Research*, 111, 129-141.

Valls, V., M. Laguna, P. Lino, A. Pérez and S. Quintanilla, 1999, Project Scheduling with Stochastic Activity Interruptions, Chapter 15 in Weglarz, J. (editor), *Project Scheduling – Recent Models, Algorithms and Applications*, Kluwer Academic Publishers, 333-353.

Wang, J.R., 1999, A Fuzzy Set Approach to Activity Scheduling for Product Development, *Journal of the Operational Research Society*, 50, 1217-1228.

Whitehouse, G.E., 1973, *Systems Analysis and Design Using Network Techniques*, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Wiest, J.D. and F.K. Levy, 1977, *A Management Guide to PERT/CPM: with GERT/PDM/CPM and Other Networks*, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Wollmer, R.D., 1985, Critical Path Planning under Uncertainty, *Mathematical Programming Study*, 25, 164-171.

Wu, S.D., R.H. Storer and P.C. Chang, 1993, One Machine Rescheduling Heuristics with Efficiency and Stability as Criteria, *Computers and Operations Research*, 20, 1-14.